Reconocimiento de acciones corporales mediante aprendizaje profundo con datos sensoriales de dispositivos móviles

---

Recognition of body actions through deep learning with sensory data from mobile devices

Trabajo de Fin de Grado
Curso 2020–2021

Autores
Mercedes Herrero Fernández
Nerea Jiménez González

Director
Gonzalo Pajares Martinsanz

Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

Reconocimiento de acciones corporales mediante aprendizaje profundo con datos sensoriales de dispositivos móviles

Recognition of body actions through deep learning with sensory data from mobile devices

Trabajo de Fin de Grado en Ingeniería Software
Departamento de Ingeniería de Software e Inteligencia Artificial

Autores
Mercedes Herrero Fernández
Nerea Jiménez González

Director
Gonzalo Pajares Martinsanz

Convocatoria: *Junio 2021*
Calificación:

Grado en Ingeniería del Software
Facultad de Informática
Universidad Complutense de Madrid

5 de junio de 2021

Agradecimientos

Mercedes Herrero

En primer lugar, me gustaría agradecer a mi tutor Gonzalo, por cada consejo y sugerencia recibida durante el proyecto, por estar siempre dispuesto a atender nuestras dudas y guiarnos con su gran profesionalidad. Gracias a mi compañera Nerea por amenizar este trabajo desde el primer día. Gracias a todos los profesores que han dedicado su tiempo en resolver mis dudas y darme sus consejos para mejorar tanto a nivel profesional como personal.

También quiero agradecer a todas las personas que me han hecho más llevaderos estos años. En especial a Juan, Esther y Jairo por estar desde el principio. A las CUP por ser mi desconexión, mi fuente de energía y los hombros donde me desahogo. A David por confiar más en mí que yo misma, gracias por acompañarme de la mano y darme un empujón cuando lo necesitaba.

Para terminar, gracias a mi familia. A mis padres, por el enorme esfuerzo que han hecho para darme la enseñanza que quería, por inculcarme que con trabajo y perseverancia puedo alcanzar las metas que me proponga. A mis hermanos, por relajarme con risas siempre que necesito tomarme un respiro. A mi tía Ana, mi ángel que me cuida, gracias por enseñarme que la vida hay que disfrutarla venga lo que venga, ojalá pudiera enseñarte todo lo que he aprendido estos años.

Nerea Jiménez

Agradecimientos a todas las personas que me han ayudado día a día a seguir con la carrera. A la asociación ASCII por hacer que mi vida universitaria estuviese llena de recuerdos. A mis padres por apoyarme y darme mi espacio cuando lo necesitaba. A mi hermana por acompañarme en los momentos bajos. A mi novio por aguantarme en los momentos malos.

A mis amigos, Toni por acompañarme en muchos momentos especiales para mí, Fer por estar ahí incluso cuando yo parecía no estar, Aru por seguir ahí aún cuando parecía que no y ayudarme con Latex Y BibTex, sin ti probablemente hubiese terminado odiando este trabajo. Alv por hacer de mis primeros años de carrera los mejores, a Juan por seguir a mi lado incluso en los peores momentos. A mis chicas Belén, Ela, Silvia y V por ayudarme a sentir que no estaba sola en la facultad. Agradecer a Mario que me ayudó a realizar el logo de la aplicación y que ha sido un compañero de oro. Agradecer a todos aquellos que me hacían compañía en las actividades, Eva, Richard, David, Tomás, Rubén, gente de videojuegos que vosotros sabéis quienes sois, gracias por aquel año maravilloso de bailes. Gracias a David Pascal, que me ha ayudado en muchos momentos y ha sido mi mentor de Látex.

Gracias en especial a mi tutor del TFG, Gonzalo, por hacer que este trabajo final no se convirtiese en un infierno, si no en otro paso más de la carrera. Gracias a mi compañera Mercedes, por ayudarme a apretar cuando se necesitaba, y a relajarse cuando hacía falta. Y finalmente, gracias a todos aquellos profesores que me han ayudado a llegar aquí, con sus tutorías y su paciencia.

Resumen

En la actualidad, el uso de redes neuronales recursivas es muy frecuente, aunque no lo veamos a simple vista. Desde nuestras pulseras inteligentes hasta la aplicación más sencilla, es posible que exista una de estas redes trabajando en un segundo plano. En este proyecto mostramos cómo, con una sencilla red, se puede llegar a clasificar movimientos corporales. Para ello se propone el uso de una red neuronal del tipo LSTM (*Long Short Term Memory*), de forma que utilizando datos de aceleración procedentes de los sensores integrados en un dispositivo móvil, es posible, por un lado entrenar el modelo de red indicado para proceder, en segundo lugar a la clasificación de nuevas acciones.

Se ha diseñado una aplicación que realiza esta funcionalidad. Además, se propone una extensión en el ámbito de lo que se conoce como Internet de las Cosas (*IoT, Interner of Things*) de forma que los resultados obtenidos se cargan en un servidor remoto, esto es, en la nube, que permite el acceso e intercambio de la información almacenada, mediante la red social *Twitter*. En el planteamiento que se propone se amplía la posibilidad de clasificar movimientos mediante una red previamente entrenada o bien diseñar una red definida *ad hoc* según las acciones requeridas.

Palabras clave: redes neuronales recurrentes, clasificación, movimientos, IoT, ThingSpeak,redes LSTM, entrenamiento, aprendizaje automático, inteligencia artificial

Abstract

Currently, the use of recursive neural networks is very frequent, although we don't see it right away. From our smart bracelets to the simplest application, it is possible to find one of these networks working in the background. In this project, we show how with a simple network, it is possible to classify body movements. For this, the use of a neural network of the LSTM type (Long Short Term Memory) is proposed. It is feasible to use acceleration data from the sensors integrated in a mobile device and also to train the indicated network model to proceed to the classification of new shares.

An application has been designed that performs this functionality. In addition, an extension is proposed in the field of what is known as the Internet of Things (IoT) so that the results obtained are uploaded to a remote server, that is, in the cloud, which allows access and exchange of stored information, through the social network Twitter. The suggested come up with the possibility of classifying movements through a previously trained network or designing a defined ad hoc network according to the required actions.

Keywords: recurrent neural networks, classification, movements, IoT, ThingSpeak, LSTM networks, training, automatic learning, artificial intelligence.

Índice

Índice de figuras

CAPÍTULO 1

Introducción

El reconocimiento de acciones humanas es un campo en continua expansión gracias al desarrollo tecnológico de los dispositivos móviles, equipados con sensores de movimiento que permiten identificar actividades humanas en función del tipo de movimiento realizado por una persona equipada con un dispositivo de tal naturaleza.

El deporte o la salud son dos áreas de interés en este sentido, donde dispositivos tales como relojes de actividad, velocímetros o los propios *smartphones* son elementos dotados de la tecnología sensorial suficiente con tal propósito.

Dada la amplia difusión hoy en día de los *smartphones* entre la población en general, ha permitido enfocar el presente proyecto en el sentido de desarrollar una aplicación de detección de actividad inteligente, utilizando técnicas de aprendizaje profundo, para un *smartphone* equipado con sensores de velocidad angular, aceleración, posición, campo magnético y orientación. Simultáneamente, la aplicación se plantea con una extensión a otro de los paradigmas de máxima actualidad, como es Internet de las Cosas (IoT, *Internet of Things*).

1.1. Consideraciones generales

El Reconocimiento de Acciones Humanas, conocido a nivel internacional como *Human Action Recognition* (HAR), constituye un campo tecnológico que ha cobrado un gran auge en los últimos años gracias al desarrollo y

crecimiento exponencial de las tecnologías computacionales y en especial de las enmarcadas en el ámbito de la Inteligencia Artificial, cuyo máximo exponente en la actualidad se centra en lo que se conoce como Aprendizaje Automático (*Deep Learning*)[1]. Es precisamente bajo este paradigma sobre el que se desarrolla el presente trabajo.

Es posible encontrar en el mercado diferentes dispositivos que realizan acciones de tipo HAR en diferentes áreas, ya sea bien en el deporte o la salud, por citar dos ámbitos claros de aplicación. Además de las anteriores, y aunque probablemente enmarcadas en alguna de dichas áreas, conviene reseñar su utilidad a nivel de monitorización de acciones para determinar la actividad o inactividad, cuando se supone que ésta puede ser un riesgo en distintos segmentos de la población como puede ser la videovigilancia orientada a actividades en niños o ancianos, según el momento del día.

Existen desarrollos tecnológicos comerciales que proporcionan información específica con la que llevan a cabo procesos de tipo HAR. Los más comunes que podemos encontrar a disposición y uso entre la población son relojes de actividad, relojes inteligentes y los propios *smartphones*. Generalmente, estos dispositivos vienen equipados directamente con sensores que recogen los datos necesarios para realizar dicho reconocimiento de acciones o bien mediante conexión indirecta vía *bluetooth* u otro tipo de conexión a otro dispositivo, por ejemplo, un reloj conectado a un *smartphone*. Los datos más comunes que recogen estos sensores son relativos a la velocidad angular, aceleración, posición, campo magnético y orientación. Las acciones por reconocer se encuentran programadas con carácter general en función de los datos recibidos y de forma genérica.

Puede ser que, en el caso de relojes, no sea tan común encontrar tal dispositivo en un amplio número de personas con posibilidades de uso. En comparación, un gran porcentaje de la población dispone de un *smartphone*, lo que convierte a este dispositivo en un elemento de utilización más generalizada.

Esta es una razón principal por la que, en este proyecto, se ha optado por plantear una aplicación de uso utilizando los sensores de un *smartphone* y el propio dispositivo como elementos de soporte para captura de datos, y envío de los mismos a un servidor remoto para su procesamiento, en lo que se conoce la nube. Además de lo anterior, la justificación de esta opción estriba en la posibilidad de generar los propios datos, diseñar una estrategia de procesamiento, que en este caso está fundamentada en el *Aprendizaje Profundo*[1] y más concretamente en los modelos conocidos como *Long-Short Term Memories* (LSTM) que es la estrategia inteligente utilizada[2], y que se describen con detalle en el capítulo tres. De aquí en adelante se utiliza indistintamente, e incluso mezclada, terminología en español o en inglés, atendiendo a su mejor identificación.

En definitiva, el trabajo que se plantea consiste en el desarrollo de una aplicación inteligente con base en un *smartphone* como dispositivo habilitado para captura de datos sensoriales, que son enviados a un modelo de red del tipo LSTM, previamente entrenado y alojado en la nube, donde se realiza la predicción, y que devuelve la acción clasificada al propio dispositivo.

El modelo, además, está dotado de una funcionalidad específica bajo el paradigma IoT, por tanto, el trabajo que se presenta consiste en una aplicación inteligente en IoT. Aunque posteriormente, en el capítulo tres se describe el modelo arquitectónico del sistema con un mayor nivel de detalle y lógica de funcionamiento, en la figura 1.1 se proporciona el esquema general con sus componentes principales, estableciendo así el punto de partida del planteamiento formulado en el presente trabajo.

1. Se dispone de un procesador central convenientemente equipado donde se realiza el entrenamiento de la red LSTM, aunque también podrían ser entrenados en la nube, concretamente en Matlab Drive[3], de forma que el modelo entrenado se encuentra accesible desde un dispositivo móvil.

2. Mediante el dispositivo móvil, conectado con Matlab Drive, se capturan secuencias de datos de movimiento que permiten: a) almacenarlos en la nube, en este caso en ThingSpeak[4], que es la plataforma analítica propia de TheMathworks para IoT y b) proporcionar secuencias de movimiento al modelo LSTM entrenado para la clasificación de una determinada acción.

3. ThingSpeak recibe además los datos resultantes de la clasificación devolviendo el correspondiente mensaje vía Twitter sobre la acción identificada.

4. Los diferentes módulos se comunican entre sí mediante conexión inalámbrica, bien por internet o cualquier otro tipo de red tal como 4G y en el futuro próximo 5G.

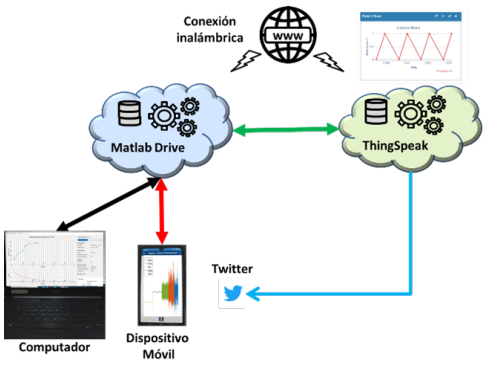

Figura 1.1: Esquema general del sistema

1.2. Estado del arte

La idea que se plantea en el presente proyecto tiene su inspiración y antecedentes en desarrollos previos, dada su constatada utilidad. Tal es el caso de la propuesta formulada por Bayat y col.[5], que utiliza un *smartphone* con acelerómetro para obtener datos de aceleración en los tres ejes X, Y y Z. Para identificar las acciones se utiliza la herramienta Weka[6] y para la clasificación de las mismas se utiliza el Perceptrón Multicapa y otras técnicas tales como Máquinas de Vectores Soporte o *logistic model trees*. En esta misma línea se encuentra el trabajo previo de Casale y col.[7] que utiliza el clasificador *Random Forest* como método de reconocimiento de las acciones. Pilataxi y col.[8], utilizan también Máquinas de Vectores Soporte con el mismo objetivo de detección de acciones de movimiento humanas.

Una aplicación de similar naturaleza se presenta en el trabajo de Pavón[9] como trabajo fin de grado durante el curso 2019-20. En este trabajo se utiliza igualmente un dispositivo móvil que captura datos relativos al movimiento, concretamente datos de aceleración en los tres ejes, que representan acciones significativas de una cierta actividad humana. Los datos, convenientemente procesados, en lo que respecta a eliminación de ruido y la aparición de datos espurios superpuestos, son enviados para su procesamiento a un servidor, implementado como una rest API de Django utilizando Python. El procesamiento se realiza mediante un modelo de red neuronal de tipo LSTM desarrollada en Matlab[10] e integrada en el mencionado servidor como un paquete compilado. La captura de datos, su envío al servidor y la recepción de los resultados se realiza mediante la correspondiente aplicación desarrollada bajo Android.

Como se ha indicado previamente, las aplicaciones HAR sirven para múltiples propósitos, dentro de ellos destaca el reconocimiento de caídas de personas, cobrando especial relevancia en el caso de personas vulnerables y/o

de avanzada edad. En este sentido Kozina y col.[11] proponen un sistema para el reconocimiento de caídas basada en la captura de datos a partir de acelerómetros instalados en dispositivos portables.

Ciertamente, la metodología basada en LSTM ha cobrado gran relevancia en HAR desde hace más de una década, de ahí que diversos trabajos hayan abordado el tema bajo esta perspectiva, tal es el caso de la propuesta realizada por Sa-nguannarm y col.[12] que llegan a demostrar que la utilización de acelerómetros en HAR mejora el rendimiento con respecto al uso de técnicas basadas en Redes Neuronales Convolucionales[1]. No obstante, conviene señalar que existen otros trabajos que proponen el uso de métodos basados en análisis de videos para HAR, tal es el caso puesto en [13]. Si bien, aunque desde el punto de vista del desempeño en cuanto a la mejora de resultados puede ser favorable frente a métodos basados en LSTM, lo cierto es que necesitan siempre que la persona realizando la acción esté situada bajo el campo de vista de la cámara, algo que no es necesario en el enfoque propuesto en este trabajo, ya que el único requisito es que la persona lleve consigo el *smartphone* con los sensores de aceleración activados. Esta es una poderosa razón por la que en este trabajo se ha optado por la opción de utilizar las técnicas basadas en LSTM para HAR, además de su uso extendido en diversas aplicaciones, complementado con el hecho de que estos dispositivos son de uso común y en continuo crecimiento en la población con carácter general.

1.3. Objetivos

El objetivo general de este proyecto es conseguir clasificar los movimientos realizados por una persona, a través de los sensores de un dispositivo móvil, utilizando para ello una red neuronal LSTM.

Como ampliación u objetivo extra, se propone implementar un modelo de red particularizada y entrenada con movimientos definidos por nosotras mismas, de forma que se pueda crear una red que clasifique aquellos movimientos deseados, y por tanto, adaptando el rango de utilidad del modelo a acciones específicas propias.

Para cumplir los objetivos previos, se plantean los siguientes objetivos específicos que debe cumplir la aplicación propuesta. Son los siguientes:

- Diseño de un módulo de captura de datos.
- Diseño de un módulo de procesamiento de datos mediante LSTM, que incluye entrenamiento y reconocimiento de acciones con los datos obtenidos.

- Diseño de un modelo de red LSTM *ad hoc*, adaptado para el reconocimiento de acciones propias.
- Diseño de un módulo bajo el paradigma IoT
- Integración de todos los módulos anteriores, incluyendo el flujo de datos entre los módulos.
- Diseño de un IHM para dar cobertura a la integración, con captura y presentación de datos y resultados.

1.4. Plan de trabajo

Al comenzar con el trabajo, se nos plantearon una serie de retos para lo cuales, nos propusimos los siguientes puntos a seguir:

- Investigación y documentación del tema general.
- Aprender conocimientos básicos de Matlab.
- Aprender conocimientos básicos de Látex.
- Estructuración de la aplicación.
- Número de módulos de la aplicación.
- Dividir las tareas entre los miembros del trabajo.
- Implementación de los módulos de la aplicación.
- Análisis de los resultados de la aplicación.
- Documentación en la memoria del transcurso del trabajo.
- Correcciones y revisiones de la memoria.
- Posible ampliación.

1.5. Estructura del documento

Los diferentes capítulos que componen este documento son:

- **Capítulo 1, Introducción**
  Consideraciones generales, estado del arte, objetivos, estructura del documento, plan de trabajo y contribuciones personales.

- **Capítulo 2, Estado del arte: métodos aplicados**
  Teoría sobre los métodos aplicados en este proyecto, relativos a conceptos tales como la inteligencia artificial, el aprendizaje automáticos, las redes neuronales recurrentes, y las redes LSTM.
- **Capítulo 3, Análisis y diseño**
  Explicación sobre la arquitectura, las tecnologías y diseño de la aplicación.
- **Capítulo 4, Resultados**
  Recursos utilizados y resultados integrados de la aplicación, relativos a los procesos de entrenamiento y clasificación del modelo.
- **Capítulo 5, Conclusiones y trabajo futuro**
  Conclusiones y trabajo futuro.
- **Bibliografía**
  Bibliografía relativa al trabajo
- **ANEXO, Uso de la aplicación**
  Guía sobre cómo utilizar la aplicación.

Tras ir realizando los puntos, finalmente se decidió realizar la ampliación, debido a que el transcurso fue positivo.

1.6. Contribuciones personales

En el desarrollo de esta aplicación se han realizado diferentes tareas, solapándose estas en algunos casos y siendo realizadas conjuntamente por ambas componentes del equipo.

1.6.1. Mercedes Herrero

Labores en investigación:

- Investigar sobre el uso del sistema de composición de textos Látex con el fin de utilizar este recurso para el desarrollo de la memoria.
- Estudio del funcionamiento de las redes LSTM.
- Investigar el uso de distintas arquitecturas de red LSTM.
- Formación en las nociones básicas del lenguaje de programación MATLAB a través de distintos cursos que pone a disposición MathWorks.
- Investigar el uso de App Designer, un entorno de desarrollo para el diseño de interfaces totalmente integrado con MATLAB.
- Capturar registros de datos propios para entrenar la red.
- Realización de distintos movimientos para comprobar el funcionamiento de la red entrenada.
- Investigar la diferencia de resultados entre una red entrenada con datos propios y la red entrenada mediante los datos HumanActivityTrain, que son datos proporcionados por Mathworks.
- Estudio del funcionamiento de las distintas aplicaciones que pone a disposición Thingspeak.
- Vinculación de cuenta de Twitter con Thingspeak para utilizar la aplicación de ThingTweet.

Gestión del proyecto software:

- Creación de un tablero Kanban mediante la herramienta Trello.
- Identificar las diferentes tareas necesarias para realizar el proyecto.
- Fijar prioridades entre las distintas tareas.
- Organizar las tareas según su función y estado.
- Establecer fechas de entrega para las tareas.
- Actualizar las tareas finalizadas.
- Identificar las dudas e impedimentos para consultar con Gonzalo.
- Establecer reuniones junto con mi compañera Nerea para ser conscientes del estado del proyecto y establecer nuevos objetivos.

Diseño y desarrollo software:

- Implementación de la conexión al smartphone para proceder a la captura de datos en vivo y clasificación de estos.
- Implementación de la moda estadística de la clasificación de datos importados desde un archivo.
- Desarrollo e implementación de la interfaz de clasificación de datos.
- Testeo de la aplicación con distintos sistemas operativos.
- Implementación para el control de errores de la aplicación.
- Creación e integración con Matlab de un canal de ThingSpeak destinado a almacenar los valores X, Y, Z de las aceleraciones de los registros capturados a través del smartphone.
- Creación e integración con Matlab de un canal de ThingSpeak que almacene los valores de la clasificación obtenidos.
- Relacionar ThingTweet con React para mandar alertas por Twitter cuando los datos de los canales cumplan unas determinadas condiciones.

Memoria:

- Desarrollo de la introducción inicial del capítulo 1.
- Desarrollo del punto 1.6.1: Contribuciones personales - Mercedes Herrero.
- Desarrollo del capítulo 2: Métodos aplicados.
- Desarrollo de los puntos del capítulo 3: 3.1 Arquitectura, 3.2 Tecnologías, 3.3.1 Entrenamiento de la red LSTM, 3.3.2 Clasificación de acciones y 3.3.3 IoT.
- Desarrollo de los puntos del capítulo 4: 4.1 Recursos, 4.2.2 Clasificación mediante la red LSTM, 4.2.3 Uso de ThingSpeak y 4.3 Análisis comparativo: entrenamiento y clasificación
- Desarrollo del ANEXO, uso de la clasificación de acciones.
- Revisiones periódicas de cada capítulo.
- Traducción al inglés del capítulo 1 y capítulo 5.

1.6.2. Nerea Jiménez

Labores en investigación:

- Funcionamiento de Látex
- Uso de Google Académico
- Investigación de cómo usar bibliografía en Látex
- Uso de BibTex
- Uso de Matlab Desktop
- Uso de App Designer, entorno de desarrollo para el diseño de interfaces integrado con Maltab
- Investigación de la creación de archivos a partir de movimientos propios
- Investigación de captura de movimientos a través de sensores
- Uso de Thingspeak
- Investigación sobre la relación de Matlab y Thingspeak

Gestión del proyecto software:

- Planificación y realización de reuniones con los miembros del equipo y tutor para valorar la progresión del trabajo y las futuras tareas a realizar
- Reparto de tareas para el desarrollo de la aplicación
- Planificación para las revisiones de la memoria con tutor y compañeros de equipo
- Planificación para una ampliación del TFG
- Planificación de corrección de errores y revisión antes de la entrega final

Diseño y desarrollo software:

- Implementación de la creación y entrenamiento de redes neuronales para la futura clasificación de acciones
- Desarrollo e implementación de la interfaz de la creación de redes neuronales

- Desarrollo de pestaña principal
- Creación del logo de la aplicación
- Pruebas del funcionamiento de la aplicación

Memoria:

- Creación de la estructura de la memoria en Látex
- Desarrollo punto 1.1 Objetivo
- Desarrollo punto 1.2 Estructura del documento
- Desarrollo punto 1.3.2 Contribuciones personales - Nerea Jiménez
- Desarrollo del punto 2.3 Redes Neuronales Recurrentes
- Desarrollo del punto 2.4 Redes LSTM
- Desarrollo del punto 3.2.1 Matlab
- Desarrollo del punto 3.3 Diseño
- Revisión y corrección del apartado 2
- Enumeración de ecuaciones
- Creación del índice de ecuaciones
- Desarrollo del punto 4.1 Recursos
- Desarrollo del punto 4.2.1 Entrenamiento de red
- Desarrollo del ANEXO, uso de la interfaz del entrenamiento de red
- Revisión y corrección del apartado 1
- Revisión y corrección del apartado 2
- Desarrollo del resumen y palabras clave
- Revisión y corrección del apartado 4

CHAPTER 1

Introduction

Human activity recognition is a field in continuous expansion thanks to the technological development of mobile devices. They are equipped with motion sensors that allow identifying human activities based on the kind of movement carried out by a person equipped with such a device.

Sports and health are two areas of interest, where devices such as activity watches, speedometers or their own smartphones are elements gifted with enough sensory technology with such purpose.

Due to the wide diffusion of smartphones among the general population, it has allowed us to focus this project in the sense of developing an intelligent activity detection application. We have used deep learning techniques where the smartphone is equipped with some sensors: angular velocity, acceleration, position, magnetic field and orientation. Simultaneously, the application is proposed with an extension to another of the topical paradigms, such as the Internet of Things.

1.1. General considerations

Human Activity Recognition (HAR) constitutes a technological field that has gained a great boom in recent years thanks to the development and growth of the computational technologies and the ones inside the field of Artificial Intelligence. Nowadays, Deep Learning is one of its main paradigms. This project is based on the development of a Deep Learning based approach.

It is possible to find different devices on the market that perform HAR

in different areas, either in sports or health, to name two clear areas of application. In addition to the above, it is appropiate to review its usefulness at the level of monitoring actions to determine activity or inactivity, when it is assumed that this may be a risk in different segments of the population, such as activity-oriented video surveillance for children or the elderly, depending on the time of day.

There are commercial technological developments that provide specific information with which they carry out HAR-type processes. The most common that we can find available for use among the population are activity watches, smart watches and the smartphones themselves. Generally, these devices are directly equipped with sensors that record the useful data to recognize the actions. The recognition can be determined through an indirect connection via Bluetooth or another type of connection to another device, for example, a watch connected to a smartphone. The most common data collected by these sensors are relative to angular velocity, acceleration, position, magnetic field, and orientation. The actions to be recognized are generally programmed based on the data received and in a generic way.

It may be that, in the case of watches, it is not so common to find such a device in a large number of people that use it. In comparison, a large percentage of the population has a smartphone, which makes this device a more widely used item.

This is the main reason why, in this project, it has been chosen to propose an application for use, using the smartphone's sensors and the device itself as support elements for data acquisition, and sending them to a remote server for processing what is known as the cloud. In addition to the above, the justification for this option is based on the possibility of generating your own data, designing a processing strategy, which in this case is based on Deep Learning[1] and more specifically in models known as Long-Short Term Memories (LSTM) which is the intelligent strategy used [2], described in detail in chapter three.

Definitely, the work proposed consists on the development of a smart app based on a smartphone as an enabled device to capture sensory data. These data are sent to a network model of the LSTM type that has been previously trained and hosted in the cloud, where the prediction is done.The action will return already classified to the device itself.

The model is also endowed with a specific functionality under the paradigm of the Internet of Things (IoT), therefore, this project consists of an intelligent application in IoT. Although later, in chapter three the architectural model is described in more in depth, the figure 1.1 provides the general scheme with its main components that establish the starting point of the approach formulated in the present work.

1. There is a central processor, conveniently equipped where the training of the LSTM network is carried out, although they could also be trained in the cloud, specifically in Matlab Drive [3], that way, the model trained is accessible from a mobile device.

2. Using the mobile device, connected to Matlab Drive, the sequences of motion data are captured and allow the following: a) to store them in the cloud, in this case in ThingSpeak [4], which is the analytical platform provided by TheMathworks for IoT and b) to provide sequences of movement to the trained LSTM model for classification of a certain action.

3. ThingSpeak also receives the data resulting from the classification by returning the corresponding message via Twitter about the identified action.

4. The different modules communicate with each other through a wireless connection, either over the internet or any other type of network such as 4G and in the near future 5G.

1.2. Objectives

The main objective of this project is to be able to classify the movements (actions) made by a person, through the sensors of a mobile device, using an LSTM neural network.

Another goal is to implement a particularized and trained network model with movements defined by ourselves. That way, a network can be created that classifies those desired movements, and therefore, adapting the range of utility of the model to actions on specific ones.

In order to meet the previous objectives, the following specific goals are proposed. They are as follows:

- Design of a data capture module.
- Design of a data processing module using LSTM, which includes training and recognition of actions with the data obtained.
- Design of an ad hoc LSTM network model, adapted for the recognition of own actions.
- Design of a module under the IoT paradigm.
- Integration of all the previous modules, including the data flow between the modules.

- Design of an HMI to cover integration, with data acquisition and presentation of results.

1.3. Workplan

When we started with this project, we found a few challenges for which, we proposed the following tasks to follow:

- Research and documentation of the general topic.
- Learn basic knowledge of Matlab.
- Learn basic knowledge of Latex.
- Structuring the application.
- Number of application modules.
- Divide tasks among job members.
- Implementation of the application modules.
- Analysis of the application results.
- Documentation in the memory of the course of the work.
- Memory corrections and revisions.

CAPÍTULO 2

Estado del arte: métodos aplicados

Como se ha indicado previamente, HAR está siendo un área de gran interés en diferentes sectores cuyo uso ha crecido exponencialmente en los últimos años. El mundo digital en el que vivimos y la facilidad de poseer dispositivos móviles de última generación, se ha convertido en una pasarela sencilla para el reconocimiento de la actividad humana. Hoy en día, es habitual encontrar en cada casa e incluso a nivel de uso personal un dispositivo con diferentes sensores (temperatura, giroscopio, campo magnético, acelerómetro, entre otros) los cuales, gracias a distintos procesos de adquisición, pueden utilizarse con facilidad para captar datos que se traducirán en movimientos corporales[14].

Todo esto no sería posible de realizar sin el estudio y avance en el desarrollo de la Inteligencia Artificial(IA). Es por ello que el reconocimiento y clasificación de acciones, que se propone en este trabajo, sería difícil de entender sin ubicarnos antes en conceptos relacionados con la Inteligencia Artificial, Aprendizaje Automático (*Machine Learning*), Aprendizaje Profundo (*Deep Learning*) y el tipo de redes neuronales LSTM que se utilizan para clasificar las acciones.

2.1. Inteligencia Artificial

La IA es la ciencia y a la vez ingeniería cuya finalidad es la construcción de máquinas con capacidad de toma de decisiones, que simulan el razonamiento inteligente humano, cuya materialización consiste en el desarrollo de

programas de ordenador dotados de las habilidades y capacidades inteligentes necesarias para realizar tal razonamiento. Por tanto, está íntimamente relacionada con la tarea de usar ordenadores para entender la inteligencia humana[15].

La IA es una de las ciencias más recientes, remontando su origen a mediados del siglo XX, que se puede definir, desde un punto de vista conceptual de alto nivel, como el estudio de las facultades mentales a través del uso de modelos computacionales. La IA no se centra únicamente en entender la naturaleza de las cosas, sino que a través de ella trata de profundizar y conseguir la creación de entidades inteligentes a nivel computacional. Estas entidades deberán ser capaces de simular la mente humana, aprendiendo de la experiencia, razonando y teniendo en cuenta variables como la incertidumbre o la inconsistencia de información para resolver problemas con un alto nivel de complejidad de una manera más eficiente y eficaz[16].

Los sistemas inteligentes son creados bajo experimentación y mejora. Este proceso nos lleva a la producción de varios principios de la inteligencia artificial de amplia aplicabilidad, por lo que se pueden utilizar en una gran variedad de áreas[17]. Puesto que su objetivo consiste en sintetizar y automatizar tareas intelectuales, da pie a una amplia gama de posibilidades que abarca desde usos académicos en demostraciones de teorías matemáticas o traductores inteligentes a usos médicos para el diagnóstico de enfermedades[18], y naturalmente el HAR, que es la propuesta formulada en el presente trabajo.

Tal y como se ha mencionado previamente, el trasvase de inteligencia a las máquinas bajo los principios de experimentación y mejora, que en combinación dan lugar al Aprendizaje Automático y dentro de éste al Aprendizaje Profundo, que se abordan a nivel conceptual en la siguiente sección.

2.2. Aprendizaje Automático

La necesidad de adquirir el conocimiento mediante una gran fuente de datos dio lugar al Aprendizaje Automático (AA), lo que se conoce como *Machine Learning* en la comunidad científica internacional. La introducción de este concepto permitió dotar a las máquinas de la capacidad para aprender según la información que procesan y así poder tomar decisiones que parecen subjetivas y por tanto propias. La efectividad y rendimiento de los algoritmos de aprendizaje automático dependen de la presentación de los datos que se les proporciona, ya que la tasa de acierto del modelo generado durante la toma de decisiones dependerá de estos[2]. Para el procesamiento de los mencionados datos se han planteado diversas metodologías y paradigmas, dentro de los cuales se sitúan las redes neuronales, como un elemento clave. En los años 80 del siglo pasado, se empezó con este paradigma computacional que pretende imitar el funcionamiento biológico del cerebro humano. Las

redes neuronales son sistemas compuestos por muchos elementos de procesamiento que representan conexiones entre neuronas creando como propiedad el aprendizaje a partir de los datos y la adaptación a los propios datos y al entorno de donde proceden éstos[16].

Así como los humanos aplicamos el conocimiento ganado por la experiencia a nuevos problemas o situaciones, las redes neuronales generalizan información extraída de datos de problemas ya resueltos para construir un sistema que toma decisiones y realiza clasificaciones[19].

Del desarrollo de las redes neuronales, surgió el Aprendizaje Profundo (AP) o *Deep Learning*. Su nombre y concepto deriva de la utilización de una arquitectura con un modelo de red neuronal de múltiples capas, ofreciendo un mejor comportamiento y resultado cuanto mayor es el número de datos de entrada. Las técnicas clásicas no originan la misma adaptación en las capas más profundas, es por ello que el AP se ha ido convirtiendo en la técnica líder del aprendizaje automático y actualmente el interés por su estudio y avance se ha disparado.

La Figura 2.1 muestra la estructura jerárquica conceptual y su nivel relacional de inclusión entre los paradigmas IA, AA y AP.

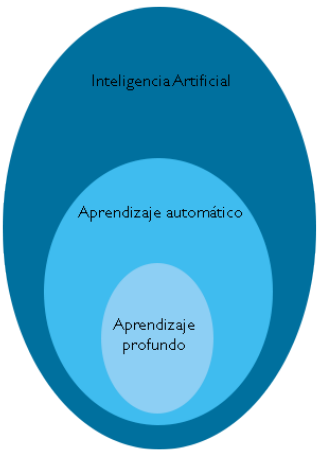

Figura 2.1: Esquema *Deep Learning*

Como se ha mencionado previamente, en este trabajo se utilizan las LSTM, que son una técnica específica dentro del AP. Se trata de una estructura específicamente diseñada para el tratamiento y análisis de secuencias temporales, razón que justifica su uso, ya que los datos obtenidos son precisamente datos de aceleración, con sus tres componentes en los respectivos ejes X, Y, Z, capturando tres valores simultáneos de las tres mencionadas componentes en cada instante de tiempo y con una duración temporal deter-

minada. El número de datos obtenidos depende de la frecuencia de captura, por ejemplo, si se especifica la frecuencia a 10Hz, se obtendrán 10 muestras por segundo, construyéndose así la secuencia temporal según la duración indicada.

Las LSTM constituyen una estructura básica dentro de lo que se conoce como redes Neuronales Recurrentes (RNN, *Recurrent Neural Network*), que establecen los fundamentos del tratamiento secuencial de datos. Por ello, a continuación se realiza una breve descripción de las RNN para abordar posteriormente la parte conceptual de las LSTM.

2.3. Redes Neuronales Recurrentes

Como se ha comentado previamente, las RNN permiten escalar secuencias de datos de grandes dimensiones que otras redes no lo permiten. Existen distintos tipos de arquitecturas para definir modelos RNN. Tal y como se muestra en la Figura 2.2 la arquitectura más simple de estas redes contienen tres capas: capa de entrada, capa oculta recurrente y capa de salida.

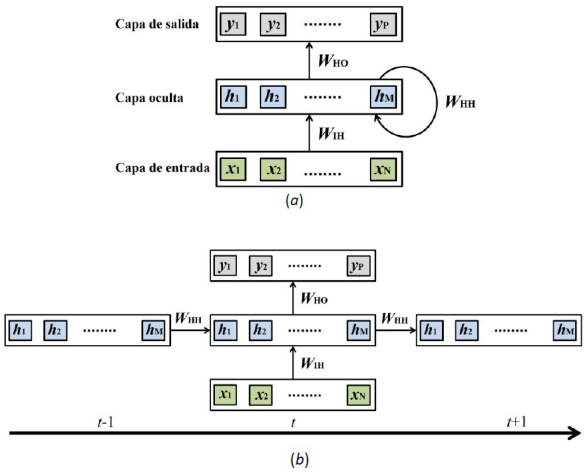

Figura 2.2: Modelo de red RNN: (a) plegada y (b) desplegada

En esta Figura 2.2 podemos ver tanto la versión plegada como la desplegada de un modelo de red RNN, con la indicación del tiempo en el caso de la figura desplegada. Cada flecha representa una conexión total entre las capas. Esta conexión se realiza a través de los correspondientes pesos representados por las matrices **W**, que son justamente las estructuras a aprender durante el proceso de entrenamiento de la red. Existen conexiones laterales y verticales. Las verticales conectan las respectivas capas de entrada, oculta y salida en determinado instante de tiempo y por tanto para una entrada dada $\boldsymbol{x}$ (en

este caso aceleraciones en los tres ejes) en el instante de tiempo t, mientras que las laterales realizan la conexión entre estructuras en distintos instantes de tiempo (t-1, t, t+1) tal y como aparece en la propia figura 2.2.

Una descripción más detallada del modelo que se utiliza en el presente trabajo es el que aparece en la figura 2.3, donde se muestra el despliegue de unidades, que se explican seguidamente. En este esquema conviene reseñar que para cada entrada, $\mathbf{x_t}$ se produce una salida $\mathbf{y_t}$, es decir, para una terna de valores de aceleración de entrada se obtiene una salida que determina o predice la acción humana concreta correspondiente a dicha salida, constituyendo por tanto la predicción del sistema HAR[1].

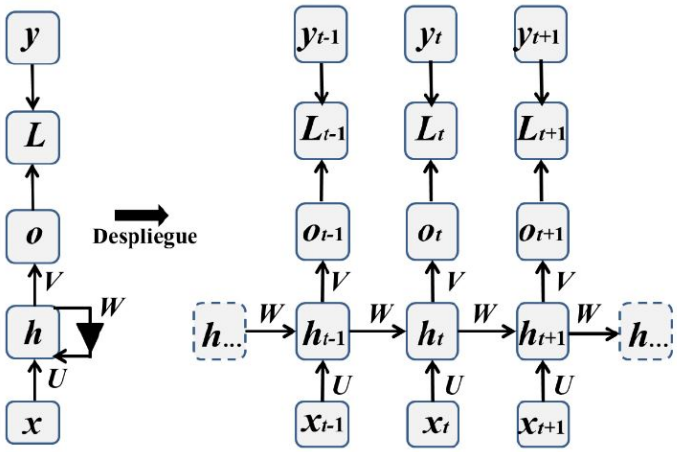

Figura 2.3: Redes recurrentes con salidas y conexiones entre unidades ocultas

Observando la Figura 2.3, que es una versión ampliada y con mayor nivel de detalle que el esquema mostrado en la Figura 2.2, en términos generales, su funcionamiento ocurre de la siguiente manera:

1. La **entrada** $\boldsymbol{x}$ corresponde a un vector de entrada, en nuestro caso un vector completo con múltiples secuencias, teniendo como valores las tres componentes de aceleración. A su vez, encontramos asociado a cada $\boldsymbol{x}$ un valor y con este su acción correspondiente (sentado, andando, corriendo, de pie o girando). En consecuencia, cada tres valores de aceleración en cada uno de sus ejes X, Y y Z (**entrada** $\boldsymbol{x}$) se asocia con una única acción (**salida** $\boldsymbol{y}$). Los **valores** $\boldsymbol{h}$ son realmente estados internos que almacenan el modelo y que se recalculan a medida que van entrando las $\boldsymbol{x}$.

2. Siguiendo con esta figura, encontramos las estructuras de datos **U**, **V** y **W** correspondientes a las matrices que contienen los parámetros de aprendizaje. Los coeficientes de dichas matrices contienen los pesos del

modelo que se establecen y ajustan durante el proceso de aprendizaje. Estos pesos determinan los valores de conexión entre las distintas unidades del modelo, de forma que una vez establecidos son determinantes para la clasificación de las acciones.

3. Los valores de las matrices de **U**, **V** y **W** se inicializan de forma aleatoria siendo posteriormente ajustados por el modelo en función de las entradas y salidas esperadas. Para cada valor $\boldsymbol{x}$ que entra se va pasando hacia arriba hasta llegar a la capa de salida donde se comprueba que la acción correspondiente al valor de entrada coincide con la $\boldsymbol{y}$ que es la acción asociada a la salida. Además, en el tránsito por cada celda, como están conectadas entre si, se obtiene información de la anterior. Esto hace que, si se genera un error porque la acción asociada a $\boldsymbol{x}$ no coincide con el valor de $\boldsymbol{y}$, se deberá evaluar una determinada función según el error cometido, que a su vez se propaga hacia atrás desde la capa de salida para reajustar los pesos de **U**, **V** y **W** hasta que la entrada $\boldsymbol{x}$ produzca la salida esperada $\boldsymbol{y}$ o al menos lo más próxima a ésta. Todos los ajustes se definen en las denominadas funciones de pérdida **L** para determinar según el modelo qué error se ha cometido. Como ejemplo, definimos la función L_t siendo esta la estima de máxima verosimilitud negativa de $\mathbf{y_t}$ dadas por los valores $\mathbf{x_1}, ..., \mathbf{x_t}$:

$$L(x_1, ..., x_t, y_1, ..., y_t) = \sum_t L_t = \sum_t logp_{modelo}(y_t|x_1, ..., x_t) \quad (2.1)$$

donde $p_{modelo}(y_t|x_1, ..., x_t)$ se define a partir de la entrada $\mathbf{y_t}$, la cual se determina a partir del vector $(\vec{y}_1, ..., \vec{y}_t)$.

4. Para lograr un alto nivel de éxito a la hora de entrenar y clasificar la acción, es conveniente tener múltiples datos para cada acción. En función del número de acciones que se desean reconocer y por tanto aprender, se establecen los vectores de salida con sus componentes necesarias.

Tal y como se muestran en la figura 2.4 se pueden establecer diferentes topologías de red según la conexión entre las distintas capas. En este trabajo se utilizará como topología de red el modelo muchos a muchos sincronizada, esto es así porque el modelo establecido es que se suministran secuencias de datos de entrada espaciadas en el tiempo que constituyen las entradas $\mathbf{x_t}$ al modelo, para cada unidad de tiempo se proporciona un valor de $\boldsymbol{x}$ y para cada $\boldsymbol{x}$ se espera un valor determinado a la salida y sincronizada en el tiempo $\boldsymbol{t}$. Por tanto, la asociación es la mencionada muchos a muchos.

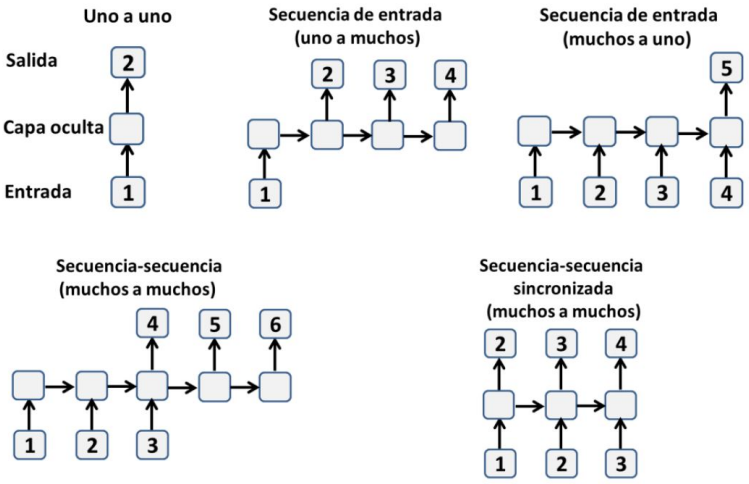

Figura 2.4: Diferentes tipologías de red

2.4. Redes LSTM

Como cualquier modelo de AP se distinguen los dos procesos característicos de aprendizaje, donde se ajustan los parámetros involucrados, y clasificación, que determina el tipo de acción realizada. Ambos se describen a continuación.

2.4.1. Modelo y entrenamiento

En el ámbito de las redes neuronales que tratan con secuencias temporales, resulta bien conocido el hecho de que durante el proceso de ajuste de los pesos del modelo por retropropagación, los gradientes se propagan hacia atrás por los diferentes estados, de forma que cuando existen muchos estados o secuencias, los gradientes tienden a desvanecerse o conducen a una explosión computacional, afectando al proceso de optimización, esto es, al ajuste. Esto es debido a que los errores propagados hacia atrás por el gradiente dependen tanto de los errores presentes como pasados en la secuencia. La acumulación de errores origina problemas en secuencias largas, de aquí el término *long-term*. Para abordar esta problemática Goodfellow y col.[20] proponen las ya mencionadas LSTM que en definitiva, ofrecen la posibilidad de establecer la información que debe conservarse o eliminarse de ahí el término *memory*. El modelo LSTM fue introducido por Hochreiter y Schmidhuber[21] con el fin de tratar la problemática expuesta, que es precisamente el modelo adoptado en el presente trabajo.

Conceptualmente las LSTM se estructuran en celdas que procesan una

secuencia de pares de entrada-salida $(\mathbf{x_t}, \mathbf{y_t})$ donde, para cada uno de ellos, la correspondiente celda LSTM toma una nueva entrada $\mathbf{x_t}$ junto con el valor oculto obtenido en la celda previa $\mathbf{h_{t-1}}$. Así se genera una estima $\tilde{\mathbf{y}}_\mathbf{t}$ para la salida objetivo $\mathbf{y_t}$ y por supuesto teniendo en cuenta la secuencia de entrada previa $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_t}$. También se genera un nuevo valor oculto $\mathbf{h_t}$ en la celda actual, que constituye de alguna manera el estado actual. Como se ha mencionado previamente, en el presente trabajo los valores de entrada en las correspondientes $\mathbf{x_t}$ son los datos de aceleración proporcionados por el sensor y por tanto, se trata de vectores tridimensionales según las tres componentes en los ejes X, Y y Z. El vector $\tilde{\mathbf{y}}_\mathbf{t}$ es la salida estimada por el modelo y proporcionada en un instante de tiempo, la cual se compara con la salida deseada u objetivo $\mathbf{y_t}$. Estas salidas se codifican de forma que en función del número de acciones a reconocer a cada salida se le asigna un valor en el vector. Por ejemplo la acción sentada se puede codificar como (1, 0, 0), la acción bailar como (0, 1, 0) y así sucesivamente. La diferencia entre la salida esperada, que por ejemplo para sentada podría ser (0.80, 0.15, 0.05) y la deseada determina el error de predicción del modelo. Si ambas coinciden, el error es nulo, lo que significa que los parámetros del modelo están bien ajustados, no requiriendo ningún ajuste. Por el contrario, si se produce un error, los parámetros del modelo deben ajustarse, constituyendo la fase de entrenamiento. Este error se establece con relación a la correspondiente función de pérdida, como por ejemplo la definida en la ecuación (2.1), tras lo cual, el error se propaga hacia atrás realizando el ajuste requerido de los pesos hasta que las salidas esperadas de las acciones se aproximen lo más posible a las predichas.

La figura 2.5 muestra el esquema general del modelo con sus correspondientes entradas y salidas, así como sus estados internos y operaciones correspondientes. En (a) se muestra un esquema con una estructura de celdas simple, mientras que en (b) la estructura interna de cada celda es más compleja, si bien en ambos casos con tres celdas interconectadas. Es importante señalar que en este modelo los estados internos $\mathbf{h_t}$ son también las salidas estimadas $\tilde{\mathbf{y}}_\mathbf{t}$. En el esquema de dicha figura, los símbolos σ_g y σ_c representan respectivamente las operaciones sigmoide y tangente hiperbólica.

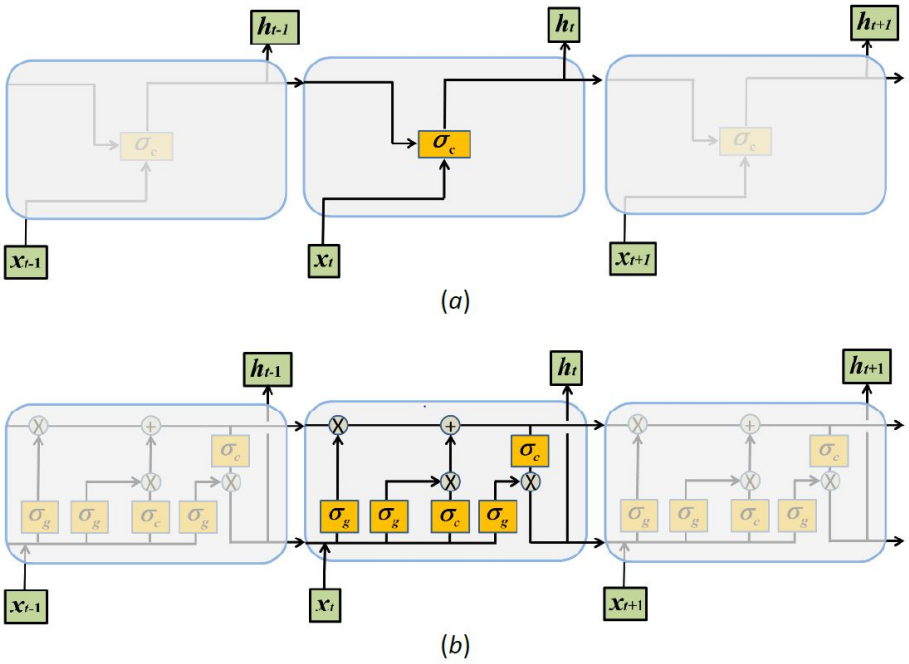

Figura 2.5: Estructura general y conexión de las celdas LSTM

Como se ve en la figura 2.5, para cada dato de entrada $\mathbf{x_t}$ se realizan una serie de operaciones:

1. En primer lugar, la capa sigmoidal, también llamada puerta de olvido o *forget gate*, es la responsable de decidir qué información eliminar del estado de la celda. Para ello, la *forget gate* utiliza la siguiente ecuación,

$$\mathbf{f}_t = \sigma_g(\mathbf{U}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{2.2}$$

2. A continuación, se debe decidir qué nueva información se va a almacenar en el estado de la celda. Para ello en primer lugar, otra capa sigmoide denominada puerta de entrada o *input gate* es la que decide qué valores se van a actualizar, apoyándose en la siguiente ecuación,

$$\mathbf{i}_t = \sigma_g(\mathbf{U}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2.3}$$

3. Tras actualizar la información, otra capa crea nuevos valores candidatos de g_t, que podrían añadirse al estado de la celda, teniendo en cuenta los pesos de las matrices $\mathbf{U}$, $\mathbf{V}$ y $\mathbf{W}$. Estos valores se crean siguiendo la ecuación descrita a continuación,

$$\mathbf{g}_t = \sigma_c(\mathbf{U}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.4}$$

4. Los valores $\mathbf{i}_t$ y $\mathbf{g}_t$ se combinan convenientemente junto con el estado de la celda previa, obteniendo de esta forma una actualización del estado total de la celda, que se puede identificar como C_t.

5. Finalmente se calcula el valor $\mathbf{o}_t$ según la ecuación 2.5, para a continuación calcular $\mathbf{h}_t$ y por tanto la salida de la celda combinando el estado actualizado de la celda con este $\mathbf{o}_t$,

$$\mathbf{o}_t = \sigma_g(\mathbf{U}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{2.5}$$

2.4.2. Parámetros y mecanismos de Aprendizaje

El proceso de aprendizaje conlleva la necesidad de definir y establecer una serie de parámetros y mecanismos de aprendizaje, entre los que destacan como de especial relevancia los siguientes:

Gradiente descendente, que es una técnica de minimización utilizada para el ajuste de los pesos involucrados en los modelos de las redes durante el proceso de retro-propagación. La función a optimizar se conoce como función objetivo y cuando se está minimizando, se le denomina también *loss function o error function*. Se trata de minimizar una función objetivo que tiene la forma,

$$J(w) = \frac{1}{n}\sum_{n}^{i=1} \nabla J_i(w)) \tag{2.6}$$

donde el parámetro w que minimiza J(w) es el que debe estimarse, constituyendo el objetivo principal del aprendizaje. J_i se asocia con la i-ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones t,

$$w(t+1) = w(t) - \varepsilon\frac{1}{n}\sum_{i=1}^{n} \nabla J_i(w) \tag{2.7}$$

De esta forma iterativa el método recorre el conjunto de datos de entrenamiento y realiza la actualización anterior para cada muestra de entrenamiento. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo consiga la convergencia. Estas muestras así seleccionadas constituyen lo que se denomina *batch*, como se describe seguidamente, a la hora de seleccionar los parámetros de entrenamiento. En este sentido, es habitual utilizar exactamente la técnica conocida como Gradiente Descendente Estocástico (SGD, *Stochastic Gradient Descent*) [22][23] o alguna de sus variantes tal como *Adam*, cuyo nombre deriva de *Adaptive Moment Estimation*[24].

Batch *size*: durante el proceso de actualización de los pesos de la red, mediante el cálculo del error y su retro-propagación a través de la técnica del gradiente descendente se dispone de N datos de entrenamiento en un proceso iterativo. Esto significa que al buscar el mínimo se realizan una serie de pasos, siendo el objetivo de cada paso ajustar lo mejor posible los pesos. La dimensión de un lote (*batch*) define el número de datos utilizados antes de llevar a cabo una actualización de los pesos de la red en el modelo.

***Epochs* e iteraciones:** define el número de veces que el conjunto total de muestras son procesadas por el algoritmo de optimización. Esto significa que cada muestra del conjunto N ha tenido una oportunidad de actualizar los pesos en cada *epoch*. También se puede fijar un número máximo de *epochs* para detener el proceso de entrenamiento cuando se alcanza dicho número, lo cual puede ser útil en el caso de que no se llegue a alcanzar la convergencia, esto es, que el error no sea aceptable.

Razón de aprendizaje (*learning rate*): determina la rapidez con la que la red actualiza o ajusta los pesos involucrados. En el diseño propuesto se establece una razón de aprendizaje variable, de forma que sobre la razón fijada inicialmente cada cierto número de *epochs* se aplica un determinado factor de reducción.

Método de optimización: es el método aplicado para realizar la minimización de la función de coste.

Softmax: consistente en proyectar los valores de la salida en la capa final al rango [0,1], proporcionando así una especie de probabilidad de pertenencia a cada una de las clases para una imagen de entrada dada. Su función se define según la siguiente expresión.

$$softmax(\mathbf{x})_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \tag{2.8}$$

2.4.3. Clasificación

Una vez finalizada la fase de entrenamiento, y con los pesos actualizados se está en condiciones de reconocer nuevas acciones. Centrándonos en el objetivo del presente trabajo, estas redes nos permiten clasificar una secuencia de movimientos que serán obtenidos a partir de los datos suministrados por los sensores de un dispositivo móvil. Como se ha indicado previamente, en cada instante de tiempo t se define un vector de entrada que representa las aceleraciones, de forma que a cada secuencia se le asocia una clase $y_t \equiv c_i$, siendo i el número de clases definidas. En nuestro caso estas clases corresponderán a movimientos convenientemente etiquetados como por ejemplo, $c_1 \equiv$ *sentada*; $c_2 \equiv$ *corriendo*; $c_3 \equiv$ *bailando*. La capa de salida de la red tiene la dimensión de un vector de etiquetas asociado a cada clase, es decir,

cada componente de este vector representa la probabilidad de un tipo de movimiento definido por la correspondiente clase, tal y como se ha indicado previamente. Con las salidas así obtenidas y teniendo en cuenta las salidas esperadas, según la acción de entrada convenientemente etiquetada, se puede incorporar como nueva muestra de entrenamiento para un proceso posterior, de esta forma se genera el correspondiente genera el correspondiente error, procediendo a la propagación del error hacia atrás para actualizar de nuevo los pesos en las matrices **U**, **V** y **W**.

CAPÍTULO 3

Análisis y diseño

En este capítulo se plantea el desarrollo de la aplicación inteligente para su uso con un dispositivo móvil de tipo *smartphone*. Para ello, en primer lugar, se expone la arquitectura de la aplicación, que incluye los módulos operativos de que consta. En segundo lugar, se describen las tecnologías utilizadas, así como sus ventajas e inconvenientes para su elección. Finalmente, en tercer lugar, se aborda el diseño de la aplicación en base a las consideraciones derivadas de la arquitectura planteada y las tecnologías analizadas.

3.1. Arquitectura

Como continuación del modelo introducido en la figura 1.1, seguidamente se describe la arquitectura del modelo planteado desde el punto de vista lógico y operativo.

Para el desarrollo e implementación de este proyecto se diferencian tres módulos básicos que permiten el uso de las diferentes técnicas citadas en los capítulos previos. Estos módulos se ven diferenciados en el esquema de la figura 3.1 representando la arquitectura de la aplicación y las tecnologías que se utilizan para la conexión y transmisión de datos.

En este sentido se observan las líneas de conexión existentes en los distintos módulos de suerte que existen comunicación y transferencia de entre ellos a distintos niveles, a través del dispositivo móvil y mediante el uso del espacio en la nube.

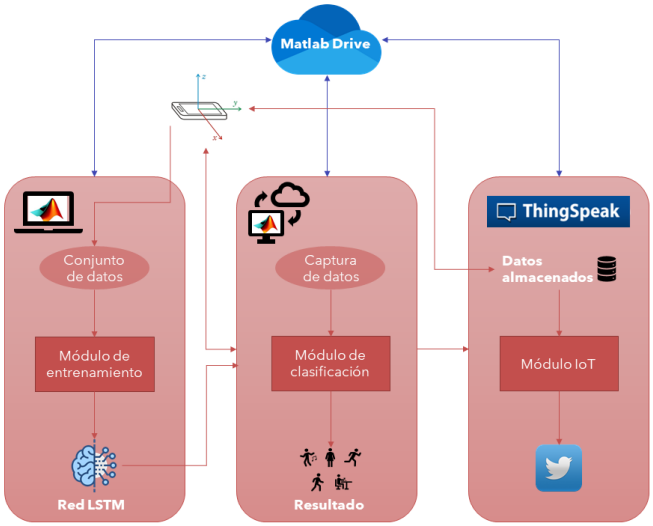

Figura 3.1: Esquema arquitectura

El proceso de funcionamiento del sistema en su conjunto y atendiendo a la arquitectura de la figura 3.1 es el que se describe a continuación.

- **Módulo de entrenamiento**

  1. El dispositivo móvil captura los datos de aceleración necesarios etiquetados según la acción que les corresponde. Estos datos se encuentran convenientemente almacenados en un fichero de datos asociada.
  2. Se dispone de un modelo de red LSTM, convenientemente configurado.
  3. Con los datos disponibles y el modelo LSTM se procede al entrenamiento de la red en el Matlab de escritorio, tras lo cual se genera el correspondiente modelo entrenado. Dicho modelo queda visible a través de Matlab Drive para el módulo de clasificación.

- **Módulo de clasificación**

  1. El dispositivo móvil captura los datos de aceleración con el objetivo de identificar la acción correspondiente, que son recibidos en este módulo, el cual a su vez accede al modelo de red LSTM.
  2. Con ello se obtiene el resultado de la clasificación de la acción, cuyo proceso se lleva a cabo a través de Matlab OnLine, que es el acceso que se realiza a través del dispositivo móvil a Matlab Drive.

3. El resultado de la acción se envía a ThingSpeak para su almacenamiento y procesamiento mediante el módulo IoT.

- **Módulo IoT**

1. Recibe el registro de los sensores junto con el resultado de la clasificación de la acción y almacena los datos convenientemente.
2. Cuando el canal lleva un tiempo sin recibir información, este envía un tweet indicando que el usuario lleva tiempo sin realizar ningún tipo de movimiento.
3. En el momento en el que se inserta en el canal la clasificación obtenida se envía un tweet referenciando al tipo de movimiento que se ha realizado.

3.2. Tecnologías

Para implementar estos tres módulos de la aplicación se realizó un estudio de las posibles tecnologías a utilizar, entre ellas:

- **Tensorflow:** Librería de código abierto desarrollada por Google Brain Team para construir y entrenar redes detectando y descifrando patrones. Nos plantemos su uso al ser la librería más extendida, sin embargo su alto nivel de complejidad hizo que descartáramos esta tecnología dadas las características del proyecto.

- **PyTorch:** Es un paquete de Python de código abierto que utiliza programación de tensores optimizada para el aprendizaje profundo permitiendo agilizar los cálculos numéricos haciendo uso de la GPU. Nos llamó la atención ya que aún siendo una tecnología reciente, su uso se ha visto disparado gracias al hecho de ofrecer una cierta complejidad con grandes ventajas. Se descartó debido a la necesidad de integrar diferentes módulos con despliegues de servicios en plataforma en la nube, tal como Google Colab[25] u otros, requiriendo un alto nivel de manejo de Python, que desbordaba los objetivos planteados en el proyecto.

- **Matlab:** Este software ofrece un entorno de trabajo con un lenguaje de programación de alto nivel para aplicaciones en el cálculo numérico. Además posee una extensa biblioteca de herramientas que facilitan la integración de distintas aplicaciones científicas. Se optó por la elección de este software dada la gran cantidad de contenidos de ejemplo que ofrece MathWorks y el libre acceso a la licencia que ofrece la Universidad Complutense de Madrid a los alumnos.

- **Thingspeak:** Es una plataforma que permite recoger, almacenar datos de sensores en la nube, y desarrollar aplicaciones IoT. Se optó por trabajar con esta plataforma ya que ofrece aplicaciones que permiten analizar y visualizar datos en MATLAB, y también enviar datos desde MATLAB a esta plataforma. También permite trabajar con redes sociales, utilizando los datos proporcionados, de una forma fácil y sencilla.

3.2.1. Matlab

Matlab ofrece la posibilidad de trabajar utilizando Matlab Cloud, proporcionando acceso instantáneo a Matlab Online y Matlab Drive. Gracias a Matlab Drive, podemos conectar los tres módulos de la aplicación y trabajar utilizando una ubicación común de almacenamiento en la nube a la cual se accede desde cualquier tipo de Matlab que se esté utilizando, tanto Mobile (que es la aplicación específica para dispositivos móviles), como Online o Matlab escritorio. Sin embargo, esta es solo una de las muchas prestaciones que nos ofrece. Otros servicios que determinaron nuestra elección como tecnología para desarrollar el proyecto son los siguientes:

- **Análisis de datos:** permite explorar, procesar, modelar y visualizar datos.
- **Gráficas:** facilidad de visualizaciones gráficas en relación a los datos y resultados obtenidos.
- **Desarrollo de algoritmos:** permite desarrollar algoritmos de forma mucho más rápida que en otros lenguajes como C, C++ o Fortran.
- **Creación de apps:** Mediante App Designer ofrece una alternativa para diseñar interfaces de usuario de forma rápida y sencilla.
- **Uso de MATLAB con otros lenguajes:** es fácilmente integrable con otros lenguajes como Java, Phyton, C++ entre otros.
- **Cálculo en la nube:** permite realizar la ejecución de procesos en entornos de nube, desde MathWorks Clouds hasta redes públicas, como AWS y Azure.

Otra de las prestaciones más importantes utilizadas en Matlab es la mencionada APP Matlab mobile, existe un toolbox que da soporte a la conexión entre los sensores del teléfono el cual permitirá capturar los datos que se utilizan bien para el entrenamiento o la clasificación. Además, Matlab mobile se puede conectar fácilmente a Matlab Drive, de forma que se comparten los datos de forma inmediata e instantánea.

Conectando con la arquitectura integrada de los módulos de la figura 3.1 y su descripción, el flujo de datos sigue el orden que se establece a continuación.

Para el desarrollo e implementación de la aplicación se hizo uso de tres versiones diferentes de Matlab: Matlab Desktop, Matlab Online y Matlab para dispositivos móviles.

En un primer lugar se realiza la recogida de datos a través de Matlab para dispositivos móviles. Estos datos se almacenan en la carpeta conjunta que comparten Matlab y Matlab Online. Cogiendo los datos de esta carpeta, desde Matlab Online tenemos dos opciones, o bien entrenar una red, o bien clasificar los datos recogidos.

Mientras que, tanto el entrenamiento de red como la clasificación de datos realizada desde un archivo se puede realizar desde Matlab Destkop o Matlab Online, la clasificación en tiempo real sólo se puede llevar a cabo desde Matlab Online, debido a que Matlab Destkop no soporta la lectura de datos que se realiza directamente de los sensores.

3.2.2. Thingspeak

Thingspeak es una plataforma de servicio de análisis de IoT que permite agregar, visualizar y controlar flujos de datos en tiempo real en la nube. La posibilidad de ejecutar código de Matlab en Thingspeak permite llevar a cabo análisis online y el procesamiento de datos que llegan en tiempo real. Algunos de los servicios que ofrece ThingSpeak son:

- Creación de canales para almacenar datos.
- Fácil configuración de dispositivos para enviar datos a ThingSpeak utilizando protocolos comunes de IoT.
- Visualizar datos procedentes del sensor de movimiento en tiempo real.
- Utilizar Matlab para dar un sentido a los datos IoT.
- Ejecutar análisis de IoT automáticamente basando en horarios o eventos.
- Actuar automáticamente sobre los datos y comunicarse con servicios de terceros concretamente a través de la red social Twitter.

3.3. Diseño

La aplicación presenta dos funcionalidades diferentes, a saber: *a)* el entrenamiento de la red LSTM, ya sea propia o default, y *b)* la clasificación

de acciones ya sea a partir de un archivo o en tiempo real. En el Anexo se proporcionan más detalles concernientes al proceso de manejo y ejecución de las distintas opciones de la aplicación, si bien desde el punto de vista de manual o guía de usuario.

En la figura 3.2 podemos ver las pantallas relacionadas con las acciones mencionadas anteriormente.

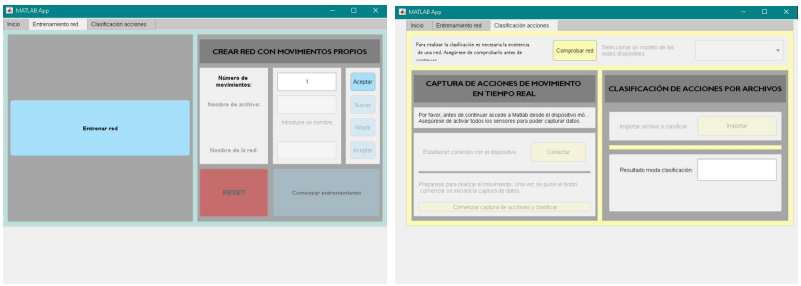

Figura 3.2: Pantallas aplicación

3.3.1. Entrenamiento de la red LSTM

Este módulo permite realizar el proceso de entrenamiento de una red neuronal de tipo LSTM para reconocer la acción desarrollada por el usuario, utilizando Matlab para ello. La red resultante entrenada será la empleada en el siguiente módulo de la aplicación. Se proporciona la opción de utilizar datos propios o un conjunto de datos por defecto procedentes de siete series de datos de sensores donde cada secuencia corresponde a la lectura del acelerómetro en tres direcciones diferentes. Independientemente del tipo de datos que se opte por seleccionar, la arquitectura de la red LSTM y sus opciones de entrenamiento son siempre las mismas, diferenciándose únicamente el número de clases utilizadas para la creación de la red.

El diseño de la arquitectura del modelo de red consta de un tamaño de secuencias de entrada de 3 elementos que corresponden a los ejes X, Y, Z; la capa LSTM posee 200 unidades ocultas generándose la secuencia completa. Finalmente, se incluye una capa completamente conectada del tamaño correspondiente al número de clases, 5 en el modo default y el número de archivos importados en el caso de entrenar la red con datos propios. Esta última capa va seguida de una capa *softmax* y una capa de clasificación. En las opciones de entrenamiento se especifica el algoritmo de optimización Adam[24] con un número de ejecuciones de 60.

El flujo de datos en el módulo está ligado al tipo de entrenamiento que se quiere realizar. Si se opta por entrenar la red por defecto los datos serán procesados por el computador donde se ejecutarán todos los procesos necesarios para realizar el entrenamiento. Sin embargo, si se quiere entrenar una red

propia, se necesita, además del computador, el uso de un dispositivo móvil para captar el registro de datos en relación con los ejes X, Y, Z mediante los sensores de aceleración, velocidad angular, campo magnético y orientación. Si bien, sólo se utilizan los datos correspondientes a aceleración.

3.3.2. Clasificación de acciones

El objetivo de este módulo es dar la oportunidad al usuario de clasificar acciones de movimiento. Para clasificar la acción es necesario capturar datos previamente y disponer de una red neuronal entrenada. La red LSTM se obtendrá del módulo de entrenamiento mientras que la captura de datos se realizará a través de la conexión con los sensores de un dispositivo móvil. Por consiguiente, es requisito fundamental disponer de un *smartphone* con acceso a Matlab Mobile para enviar el registro de los sensores a Matlab y haber realizado cualquiera de los entrenamientos que se permiten en el módulo anterior.

Existe un impedimento al utilizar Matlab Desktop, puesto que no admite la captura de datos desde los sensores del dispositivo al no permitir establecer la conexión con este. Sin embargo, Matlab Online sí realiza dicha conexión y por tanto es posible acceder en tiempo real a los datos de los sensores del *smartphone*.

Con el fin de poder ofrecer una alternativa si no se tiene acceso en ese momento a un *smartphone*, se ha desarrolado una segunda versión del módulo permitiendo importar datos previamente capturados para obtener su clasificación y moda como variable estadística de los resultados. Como en esta opción no es necesario realizar la conexión con el smartphone, la operación se puede ejecutar desde Matlab Desktop.

3.3.3. IoT

Este módulo se crea por la necesidad de llevar un historial completo de los registros del usuario y reaccionar de una determinada forma según los resultados. Todo ello se consigue gracias a las funcionalidades que proporciona ThingSpeak. Por una parte, se genera una copia de seguridad a medida que se utiliza la aplicación guardando en un canal los valores X, Y, Z de la aceleración capturada de los sensores del smartphone junto con la clasificación que proporciona el módulo anterior. A su vez, se hace uso de dos aplicaciones de comportamiento que facilita ThingSpeak:

1. ThingTweet: Vincula una cuenta de Twitter para poder mandar alertas que se configurarán según una condición establecida en los datos del canal.

2. React: Trabaja conjuntamente con ThingTweet escribiendo un *tweet* en la cuenta vinculada si no se recibe ningún dato en 60 minutos. Esta alerta se reitera de forma recurrente cada hora, en caso de seguir sin recibir datos se repetirá el tweet indicando la situación de inactividad. Por otro lado, se mandará otra alerta indicando el tipo de actividad cada vez que se inserten datos en el canal. En la figura 3.3 se aprecian las distintas alertas establecidas junto con el mensaje que se escribe en la cuenta de Twitter vinculada haciendo referencia a las condiciones explicadas anteriormente.

Todos estos datos provienen del uso del módulo de clasificación, por lo que ambos módulos funcionan conjuntamente compartiendo el flujo de datos.

Reacts using ThingTweet

Name	Message	Last Sent	Twitter Account
Inactividad	Parece que te has enganchado a una serie. Llevas mucho tiempo sentado, deberías moverte.	2021-05-27 16:00	DappHar
Sentado	Parece que estás sentado	2021-05-25 18:05	DappHar
De pie	¿Estás esperando el bus?	2021-05-25 18:11	DappHar
Andando	Si sigues a este ritmo llegarás a los pasos diarios recomendados.	2021-05-25 18:09	DappHar
Corriendo	¿Pretendes apuntarte a la San Silvestre? Sigue así	2021-05-25 18:11	DappHar
Bailando	Las discotecas están cerradas pero lo vas a petar cuando abran!	2021-05-25 18:08	DappHar

Figura 3.3: Alertas de la aplicación React

CAPÍTULO 4

Resultados

En este capítulo se analizan los resultados de los procesos integrados en la aplicación, distinguiendo fundamentalmente entre los procesos de entrenamiento y clasificación. Previamente se describen los recursos utilizados, tanto a nivel de dispositivos como en relación con el tipo de datos utilizados y capturados.

4.1. Recursos

Como se ha comentado anteriormente, es necesario tener a disposición dos tipos de recursos para utilizar la aplicación, un *smartphone* y un ordenador, ambos con conexión inalámbrica. Se han utilizado dos *smartphones* Android para la recogida de datos, y para testear distintos tipos de sistemas operativos. Además, se ha hecho uso de una tablet, donde se ha comprobado que la aplicación es totalmente compatible con IOS.

Para el entrenamiento de la red y ejecución de la aplicación se han utilizado dos ordenadores que, aunque difieren en la velocidad de la CPU, el entrenamiento del modelo de red LSTM viene a ser equivalente. En la tabla 4.1 se muestran los tipos de dispositivos móviles y ordenadores utilizados, junto con el modelo, el sistema operativo, la CPU, la memoria RAM y el procesador.

Dispositivos					
Tipo	Modelo	SO	CPU	RAM	Procesador
Smartphone	Redmi Note 8T	Android 10 QKQ1.200114.002.	Octa-core Max 2.2GHz	3GB	Snapdragon 665
Smartphone	Mi 8 Lite	Android 10 QKQ1.200114.002.	Octa-core Max 2.2GHz	4GB	Snapdragon 660
Tablet	Ipad mini 2	IOS 12.4.5	1.30MHz	1GB	Apple A7
Ordenador	MSI CX61 2QF	Windows 10 Pro 64 bits	2.60GHz	8GB	Intel Core i5-4210M
Ordenador	Lenovo ideapad z500	Windows 10 Home	2.10GHz	16GB	Intel Core i7-3612QM

Tabla 4.1: Recursos utilizados

Otro aspecto relevante en el uso de recursos es el uso de *HumanActivity*[26][27]. Disponemos de este *dataset* gracias a que Matlab lo ofrece para el entrenamiento de la red para clasificar actividades. Este *dataset* contiene 24.075 observaciones de 5 actividades: sentado, de pie, andando, corriendo y bailando. Cada observación tiene 60 características extraídas del dato de aceleración, medido por los sensores de aceleración de un dispositivo móvil.

La figura 4.1 muestra la representación gráfica de una secuencia de datos de aceleración a lo largo del tiempo precedente del conjunto de datos original *HumanActivity*, donde se aprecian claramente las oscilaciones correspondientes a los movimientos de alta variabilidad en la aceleración, como son las de "Dancing" y "Running" frente a las de baja variabilidad como "Sitting" y "Standing". En el caso de "Walking" las variaciones son de naturaleza intermedia.

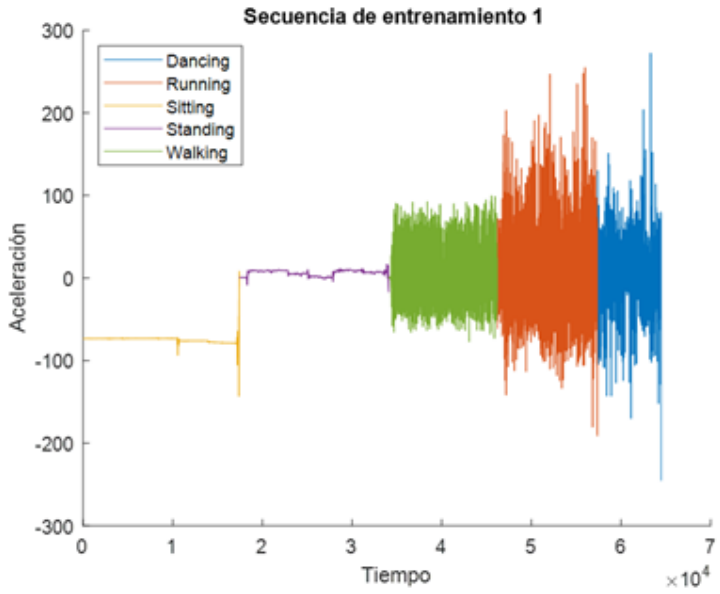

Figura 4.1: Representación gráfica de una secuencia de datos de aceleración

Por otra parte, además de los datos procedentes de *HumanActivity* se han utilizado datos propios para definir acciones específicas. Esto muestra la flexibilidad de la aplicación desarrollada y su posibilidad de adaptación para

el reconocimiento de actividades *ad hoc*, lo que otorga un valor relevante añadido a la aplicación.

Durante el proceso de entrenamiento del modelo es práctica habitual seleccionar una fracción de los datos disponibles para el entrenamiento y otra parte para validación, a veces una tercera para test. Por ejemplo 70 % para entrenamiento y 20 % para validación y 10 % para test, siendo diferentes en cada uno de los subconjuntos

4.2. Resultados integrados de la aplicación

Tras el lanzamiento de la aplicación, lo primero que aparece es la pestaña de Inicio.

Esta pestaña proporciona unas breves instrucciones para comenzar a utilizar la aplicación, como se puede ver en la figura 4.2.

Figura 4.2: Pestaña de Inicio

Las pestañas a las que podemos acceder son concretamente Entrenamiento red y Clasificaciones, que se describen a continuación. Ambas se corresponden con sendos procesos asociados a la red LSTM. En el Anexo se proporcionan detalles adicionales desde el punto de vista de un manual o guía de usuario, pero que en cualquier caso son también clarificadores del proceso de ejecución del modelo de aplicación con sus diferentes módulos integrados.

4.2.1. Entrenamiento de red

En esta pestaña podemos diferenciar dos recuadros, el primero con un botón en el que se puede leer "*Entrenar red*", y el segundo con un mayor

número de opciones, tal y como se muestra en la figura 4.3. A continuación se detalla cada uno de los procedimientos en relación a ambas opciones.

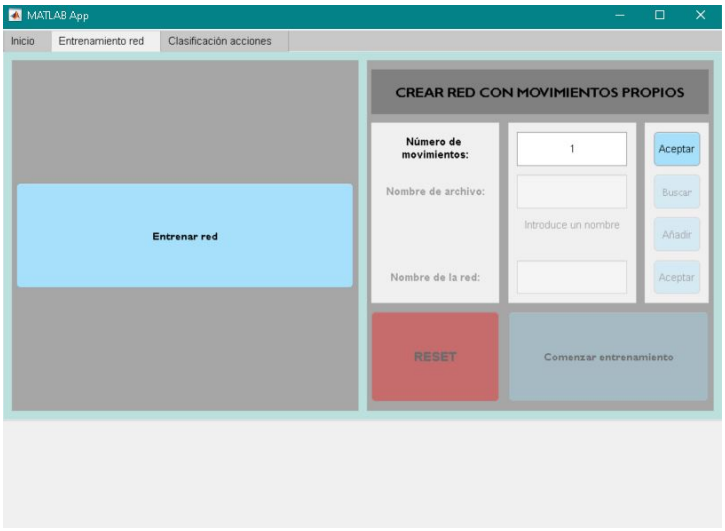

Figura 4.3: Pestaña Entrenamiento de Red

El primer botón, al ser pulsado, procede a entrenar una red *default*, con datos almacenados en la aplicación procedentes del conjunto de datos HumanActivity. Como se ha indicado previamente la red está diseñada para reconocer cinco tipos de movimientos: "Sitting", "Standing","Dancing", "Running" y "Walking". El entrenamiento de esta red, por la gran cantidad de datos procesados, del orden de veinticuatro mil, puede tardar de 2 a 4 horas, dependiendo de las características del ordenador donde se ejecute.

El segundo recuadro activa una serie de campos para introducir valores correspondientes al diseño específico del modelo de red. Los campos se podrán ir completando según se rellene de manera correcta el anterior. Como se ha indicado previamente, estos datos se utilizarán para crear una red a partir de movimientos propios, por lo que el tiempo de entrenamiento estará ligado a la cantidad de datos de cada movimiento recopilado. Por otra parte, conviene señalar que cuantos más datos tengamos de cada movimiento, más fiable será el modelo debido al mejor ajuste de los pesos involucrados en la red.

Como se puede ver en la figura 4.4 en la parte de la izquierda, según hemos ido rellenarlo los campos, se han ido deshabilitando, dejando solo posibilidad de completar el siguiente, hasta llegar a que se habilite el botón para comenzar el entrenamiento. Una vez pulsado este botón, aparece una ventana, que corresponde a la imagen de la derecha, en la cual se puede revisar si los datos que hemos introducido son correctos, y una vez que se pulsa OK comenzará el entrenamiento. Entre las opciones que aparecen se encuen-

tra el número de movimientos que se van a capturar, correspondiéndose con el número de clases de actividad a reconocer. También aparece el nombre del archivo donde se almacenarán los datos capturados correspondientes al movimiento a realizar. Finalmente, aparece el nombre del modelo de red así definida.

Figura 4.4: Segundo recuadro

Cuando se tienen los campos completados, se puede pulsar el botón de "Comenzar entrenamiento". Tras lo cual comenzará el entrenamiento de la red a partir de los archivos de datos indicados, y al finalizar el entrenamiento obtendremos una red propia con el nombre indicado. En la figura 4.5 se puede ver la ventana que aparece una vez pulsemos el botón de OK. Se observa la evolución del proceso de entrenamiento. En estas gráficas podemos diferenciar dos líneas. La línea azul indica la precisión (*accuracy*) del entrenamiento, mientras la roja indica la evolución de la función de pérdida (*loss*). La figura 4.6 muestra la leyenda asociada en ambos casos.

En relación con la mencionada figura 4.5 se observa que el proceso comienza con valores muy desfavorables tanto en lo que respecta a la precisión como a la validación, evolucionando hacia el valor óptimo, que es cero, en el caso de la función de pérdida, no así en la precisión que se ha estabilizado con valores alrededor del 40 % y por tanto lejos del ideal, que en este caso es del 100 %.

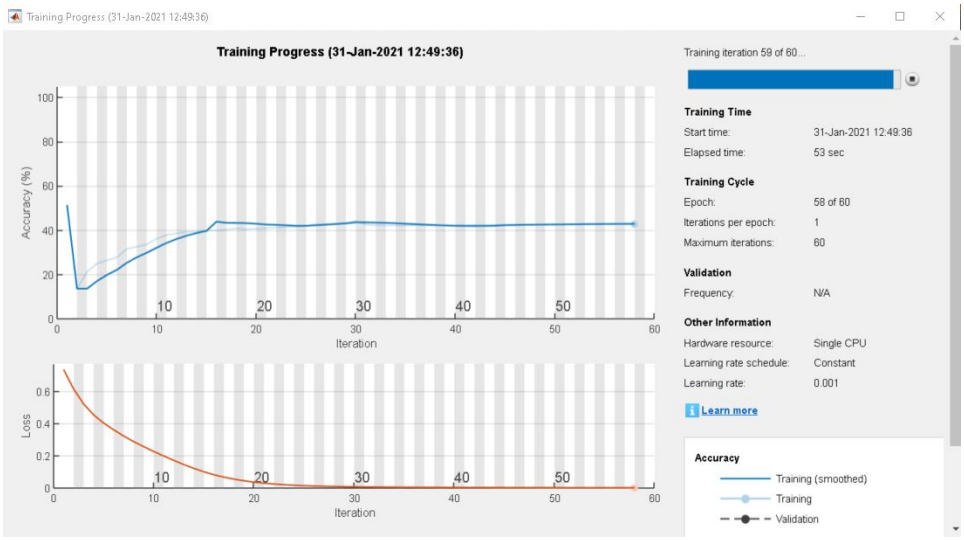

Figura 4.5: Progreso del entrenamiento de un modelo de red LSTM

Se han definido un máximo de 80 iteraciones, sobre un total de 60 *epochs* y una única iteración por *epoch*. Por otro lado, la razón de aprendizaje se ha fijado a un valor constante de 0.001 y por tanto sin disminución a lo largo de las diferentes *epochs*. Se informa igualmente sobre el tiempo transcurrido durante el proceso de entrenamiento.

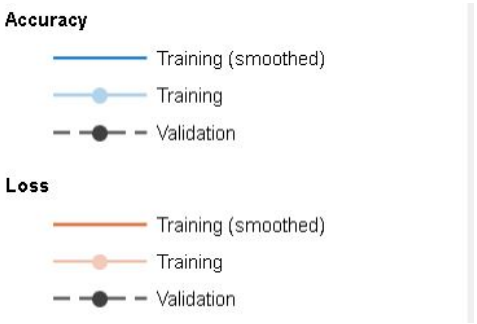

Figura 4.6: Leyenda

Durante este proceso, también se crearán otros archivos para el entrenamiento de la red. Estos se crean a partir de los datos de los movimientos recogidos, y se guardan en la carpeta de Matlab Drive denominada archivos-Train. Son utilizados para el entrenamiento de la red, y se almacenan con el mismo nombre de la red, pero añadiendo la terminación -Train.

4.2.2. Clasificación mediante la red LSTM

Como se ha mencionado en apartados anteriores en este módulo clave de la aplicación se distinguen dos secciones (captura de acciones y clasificación de acciones), como se puede apreciar en la figura 4.7.

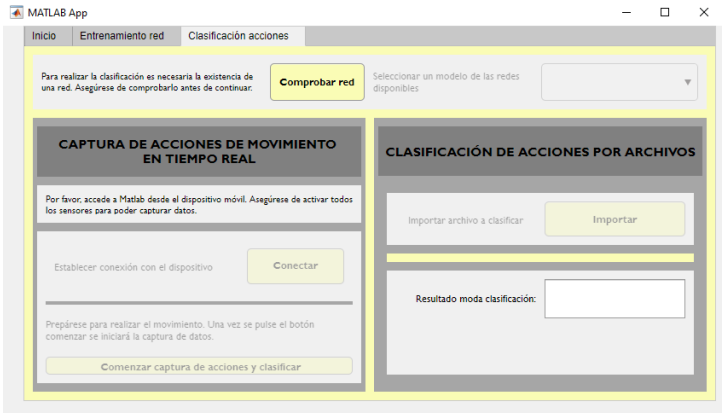

Figura 4.7: Módulo clasificación

A fin y efecto de poder ejecutar cualquiera de las dos partes es necesario comprobar la existencia de una red entrenada para poder proceder a la clasificación. Tal y como se observa en la figura 4.8, al pulsar el botón "Comprobar red" se buscará la existencia de algún modelo entrenado, habilitando un desplegable con todos los modelos encontrados o generando una ventana emergente avisando de la inexistencia de resultados.

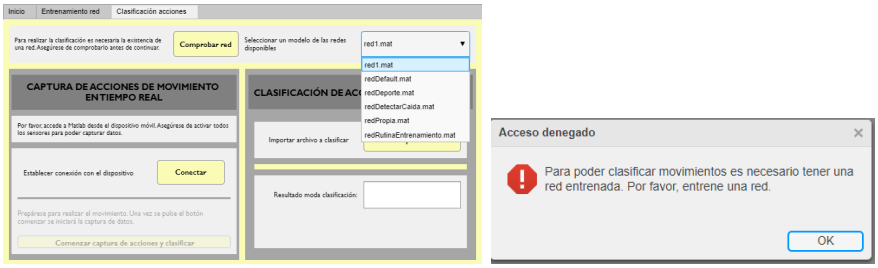

Figura 4.8: Selección de red y notificación

La parte izquierda de la figura 4.7 permite establecer la conexión a un *smartphone*, realizar la captura de datos en tiempo real y obtener la clasificación de dicha captura. Una vez finalizado el proceso se mostrarán los resultados en una ventana emergente como la mostrada en la figura 4.9 con todo el registro de los sensores y la clasificación obtenida para cada conjunto de valores X, Y, Z de la aceleración previamente registrada. A su vez,

se muestra una representación gráfica de la evolución de la aceleración y el ángulo de orientación frente al tiempo.

Obsérvese en esta figura 4.9 la distinta evolución gráfica de la aceleración y la orientación respecto de las actividades que aparecen en la tabla de la izquierda. En esta tabla aparecen cuatro tipos de actividades (*Running, Walking, Standing y Sitting*). Estas acciones implican diferentes movimientos del cuerpo según la actividad, esto se representa en la gráfica inferior, que muestra una alta variabilidad de la aceleración en su evolución a lo largo del tiempo, mientras que la orientación del sensor, representa una evolución estable a lo largo del tiempo. Esto significa que la aceleración es determinante para identificar el tipo de acción realizada, como no puede ser de otra manera, ya que son los datos de aceleración los que se utilizan para la identificación de las acciones tanto en el entrenamiento de la red como en la clasificación. No ocurre lo mismo con la orientación, ya que su estabilidad no es nada discriminante, acorde, por otra parte, con el hecho de que el modelo de red no está entrenado con este tipo de datos.

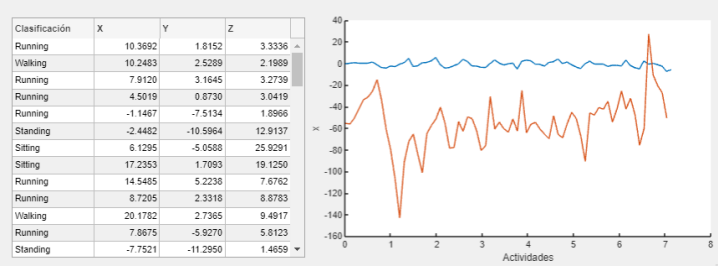

Figura 4.9: Representación de los datos capturados

En el caso de no poder realizar la conexión con el dispositivo móvil al pulsar el botón de "Conectar" se mostrará el mensaje de error mostrado en la figura 4.10.

Figura 4.10: Error al conectar el dispositivo móvil

Por otro lado, el panel en la parte de la derecha de la figura 4.8 proporciona la opción de importar archivos previamente capturados desde Matlab

Mobile sin necesidad de disponer en ese momento de la conexión con el *smartphone* y obtener la moda (valor más frecuente), como medida estadística de la clasificación de los datos. Este proceso se visualiza como muestra la figura 4.11. Mediante una ventana emergente se indica el fin de la clasificación junto con la moda estadística obtenida, a la vez que se rellena el recuadro junto a la etiqueta "Resultado moda clasificación".

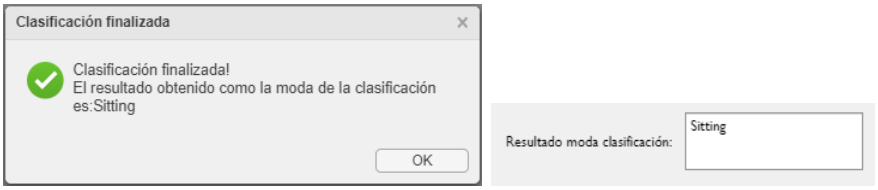

Figura 4.11: Resultado moda

Además, para dar a conocer de forma visual el resultado de la clasificación, en este caso se muestra una ventana con un diagrama de sectores como se puede apreciar en la figura 4.12, que expresa los porcentajes de acciones realizadas.

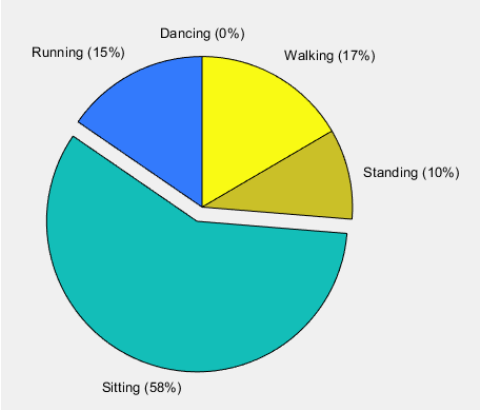

Figura 4.12: Diagrama de sectores de la clasificación

4.2.3. Uso de ThingSpeak

El proceso que se genera en segundo plano bajo el paradigma de IoT se puede visualizar en el canal donde se almacena toda la actividad de captura de acciones. Cada canal dispone de unos campos, los cuales se rellenan cada vez que se ejecuta el módulo de clasificación en tiempo real. La configuración perteneciente a este canal se detalla en la figura 4.13. Obsérvese cómo los valores correspondientes a las tres componentes de aceleración (X, Y, Z) se almacenan respectivamente en los campos 1 a 3 (Field 1, Field 2 y Field

3). En cuanto al diseño del canal, en la figura 4.14 se pueden observar los distintos campos con su representación de datos que contiene, con indicación de los intervalos de tiempo en que han sido introducidos. Aunque el campo referente al resultado de la clasificación parezca vacío, este contiene los movimientos obtenidos en la clasificación. La ausencia de elementos en la gráfica se debe a que ThinSpeak no soporta el pintado de strings, sin embargo gracias al uso de React se comprueba que la escritura de los datos es correcta puesto que cada vez que se inserta un movimiento se envía un Tweet referenciándolo.

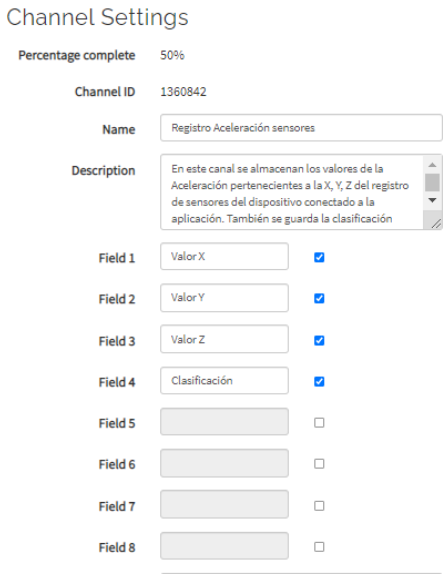

Figura 4.13: Configuración del canal de ThingSpeak

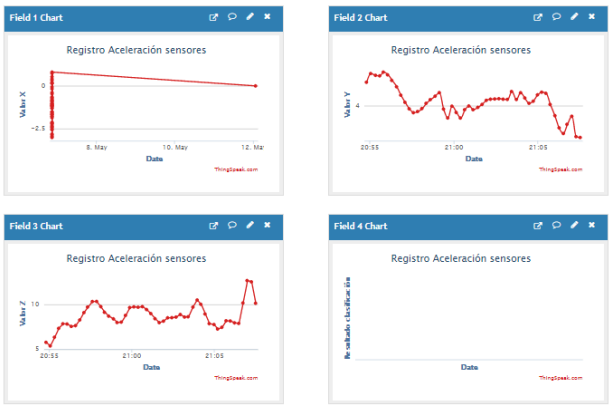

Figura 4.14: Visor canal de ThingSpeak

En lo que concierne al uso de las aplicaciones de React y ThingTweet, estas se ejecutan automáticamente cuando se cumplen las correspondientes condiciones. Como ejemplo ilustrativo, se muestra en la figura 4.15 la programación para el tipo de movimiento "Walking". En el momento en que se inserta un valor en el campo 4 del canal llamado *Registro Aceleración sensores* se comprueba si es de tipo "string" igual a "Walking". Cada vez que esta condición se cumpla se hace uso de ThingTweet mandando el mensaje estipulado a la cuenta asociada.

Figura 4.15: Programación de React

En la figura 4.16 se distinguen los distintos tweets que se envían con la programación de React para cada tipo de movimiento. Todas las alertas generadas a lo largo del desarrollo de la aplicación se pueden visualizar en la siguiente cuenta: `https://twitter.com/DappHar`.

Figura 4.16: Tweets

4.3. Análisis comparativo: entrenamiento y clasificación

Al finalizar la implementación del proyecto se procede a la evaluación los dos módulos conjuntamente para realizar un análisis de los resultados obtenidos. Con el fin de poder comparar la precisión de la red *default* con la propia se optó por realizar una recogida de datos de los mismos tipos de movimiento que admite la red *default* y entrenar con ellos la red propia. Esto permite realizar comparaciones en cuanto a los resultados de la clasificación en tiempo real y de la moda obtenida con cada una de las redes.

4.3.1. Entrenamiento

Tal y como se visualiza en las figuras 4.17 y 4.18 se aprecian tres claras diferenciaciones. En primer lugar, el tiempo de entrenamiento de la red propia es de 1 minuto frente a casi 2 horas y media en la red default. Esto se debe a la cantidad de datos importados. La segunda apreciación se distingue en lo que respecta a la precisión de las redes. En la red default la precisión no deja de crecer hasta llegar al 100 %, no obstante, en la red propia a partir de la iteración 25 se mantiene sobre el 90 % e incluso llega a bajar ligeramente antes de finalizar el entrenamiento. En último término, la gráfica correspondiente a los valores de la función de pérdida en cada iteración desciende de forma gradual en la red *default* hasta llegar a las 60 iteraciones. En cambio,

en la red propia se llega a cero en la iteración 30, lo cual indica que con esas 30 iteraciones podría ser suficiente en este caso.

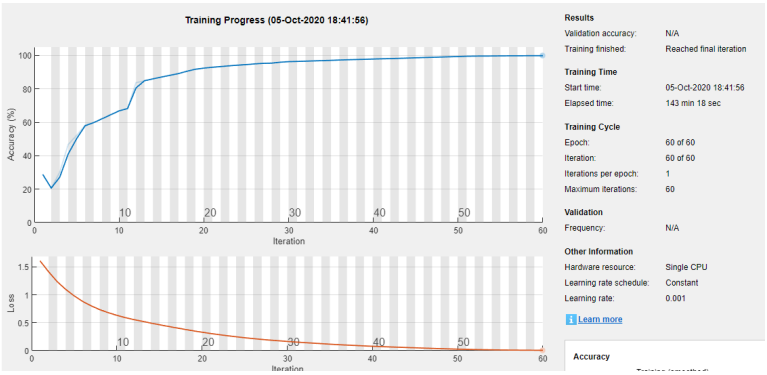

Figura 4.17: Entrenamiento red default

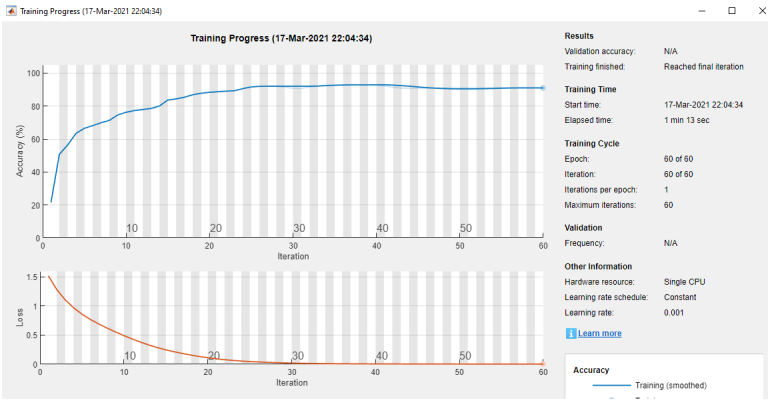

Figura 4.18: Entrenamiento datos propios

4.3.2. Clasificación

En lo concerniente a la clasificación de movimientos con miras a evaluar la tasa de acierto de cada una de las redes se capturaron un total de 3 secuencias de datos para cada uno de los 5 movimientos asumidos por la red *default*. A continuación, se muestran algunos de los resultados obtenidos frente a los esperados en función de los dos tipos de redes.

En cada una de las figuras mostradas a continuación, aparecen en la parte izquierda las clases de movimientos realizados y los valores de aceleración en los tres ejes X,Y y Z. En las figuras 4.19 y 4.20 podemos ver las gráficas relacionadas con la clasificación del movimiento Bailar(*Dancing*). De tres intentos realizados fue imposible conseguir una clasificación con el resultado Bailar para la red propia. En cuanto a la red *default* en cada uno de los tres

intentos apareció la clasificación esperada mezclando el resultado con Correr y Caminar.

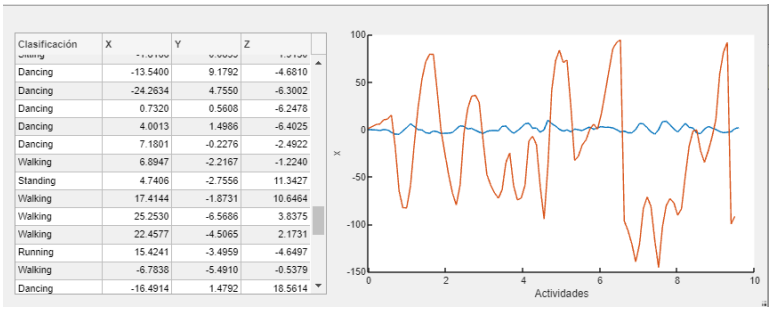

Figura 4.19: Clasificación movimiento Bailar red default

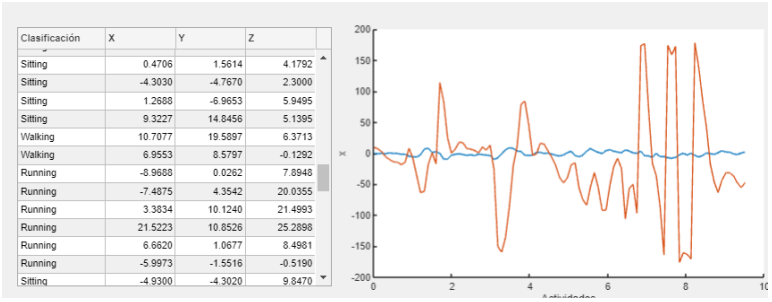

Figura 4.20: Clasificación movimiento Bailar red propia

En las figuras 4.21 y 4.22 podemos ver las gráficas relacionadas con la clasificación del movimiento Sentado(*Sitting*). En esta ocasión el 100 % de los registros fueron clasificados como se esperaba, tanto para la red *default* como para la red propia. Podríamos decir que la clasificación para *Sitting* es totalmente efectiva.

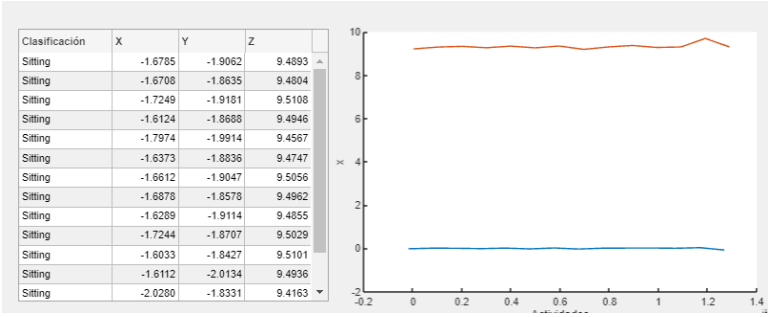

Figura 4.21: Clasificación movimiento sentado red default

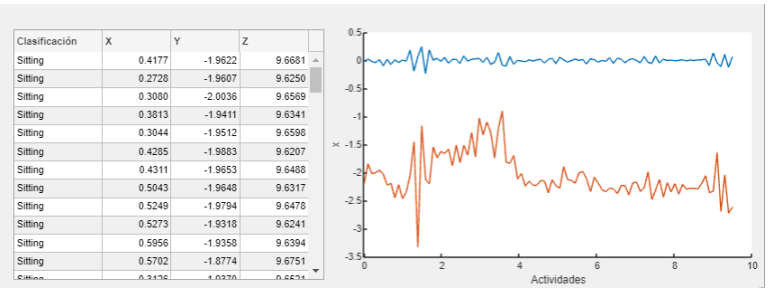

Figura 4.22: Clasificación movimiento sentado red propia

Respecto al resto de movimientos (*Standing, Walking, Dancing*) la red *default* sigue siendo, sin duda, el modelo mejor entrenado al obtener el resultado esperado en cada una de las clasificaciones. Sin embargo, la red propia interpretaba todo tipo de movimientos, sin destacar ninguno en concreto.

En pocas palabras, tal y como hemos visto en los ejemplos detallados, se comprueba que utilizando el modelo de red *default* se consigue una alta tasa de acierto en la clasificación esperada de acciones. Por consiguiente, dado que el número de datos de entrada para entrenar dicha red supera en más de la mitad a los datos de la red propia, concluimos que cuanto mayores sean los *datasets* el modelo de red estará mejor entrenado obteniendo mayor probabilidad de realizar una clasificación correcta.

CAPÍTULO 5

Conclusiones y trabajo futuro

5.1. Conclusiones

A lo largo de este proyecto, se ha desarrollado una aplicación (HAR) con el objetivo de clasificar movimientos de acciones humanas, ya sea en tiempo real o no. Este sistema hace uso de dispositivos móviles equipados con sensores de movimiento. Para el desarrollo de esta aplicación, se ha hecho uso de técnicas basadas en Aprendizaje Profundo (*Deep Learning*), concretamente el modelo LSTM para análisis de secuencias temporles e IoT.

Tras el desarrollo del sistema y la realización de pruebas, se llega a las siguientes conclusiones:

- Existe la posibilidad de conectar un dispositivo móvil con un programa, utilizando la nube como servidor remoto, y utilizando una red WiFi, para la recogida de datos en tiempo real, o para la creación de archivos de datos. (Apartado 3.1 y 3.3.1)
- Es posible obtener resultados óptimos entrenando una red con los suficientes datos, como con la red default. (Apartado 4.3)
- En el caso de entrenar una red con pocos datos, los resultados de la clasificación es menos satisfactoria. (Apartado 4.3)
- Existe la posibilidad de enviar estos resultados a un tercer destino, donde se pueden crear gráficos y vincularlo a redes sociales, en este caso, utilizando recursos y estrategias en el ámbito de IoT. (Apartado 3.3.3)

5.2. Trabajo futuro

De cara al futuro, la aplicación se puede mejorar teniendo en cuenta las consideraciones que se exponen a continuación.

Respecto a la parte de entrenamiento de red, una de las posibles mejoras sería el re-entrenamiento de una red previamente entrenada. Esto es, por ejemplo, añadir un nuevo movimiento a la red *default*, manteniendo los datos con los que ya se entrenó, y añadiendo nuevo datos para el nuevo movimiento que se quiere incorporar.

Otra posible modificación en este mismo apartado, sería la mejora de entrenamiento de las redes personalizadas. Concretamente obteniendo más datos para poder entrenar de forma más fiable las redes que se entrenan desde cero.

También como mejora del proceso de entrenamiento se puede experimentar con distintos parámetros de entrenamiento, tales como métodos de optimización, variación de la razón de aprendizaje, cambiar el número de iteraciones y *epochs*, entre otros.

Respecto a la parte de clasificación, una posible mejora consistiría en omitir la restricción del tiempo a la hora de clasificar movimientos en tiempo real. Es decir, poder recoger datos de forma continua si tener que limitar la recogida a un período de tiempo en concreto, y dejar de recogerlos cuando el usuario lo indique.

Respecto a la ampliación de ThingSpeak, una mejora posible sería el envío personalizado de tweets para el usuario. Esto es, poder personalizar a quién se envían los *tweets* para que vayan dirigidos al usuario que se desea, y no simplemente a la cuenta de la aplicación.

CHAPTER 5

Conclusions and future work

5.1. Conclusions

Throughout this project, a HAR application has been developed with the goal of classifying human actions from body movements, whether in real time or not. This system makes use of mobile devices equipped with motion sensors. For the development of this application, Deep Learning has been used, specifically the LSTM model for time sequence analysis and IoT.

After developing the system and conducting tests, the following conclusions were reached:

- There is a possibility of connecting a mobile device with a program, using the cloud as a remote server, and using a WiFi network, to collect data in real time, or to create data files. (Section 3.1 and 3.3.1)
- It is possible to obtain optimal results by training a network with sufficient data, as with the default network. (Section 4.3)
- In case of training a network with few data, the classification results are less satisfactory. (Section 4.3)
- There is the possibility of sending these results to a third destination, where graphs can be created and linked to social networks, in this case, working resources and strategies in the field of IoT. (Section 3.3.3)

5.2. Future work

Looking ahead, the application can be improved by taking the following considerations into account. Regarding the network training part, one of the possible improvements would be the re-training of a previously trained network. For instance, adding a new movement to the default network, keeping the data with which it was already trained, and adding new data for the new movement to be incorporated.

Another improvement possibility in this same section would be the improvement of training of personalized networks. Get more data to more reliably train networks that are trained right from the start. Also, as an enhacement in the training process, it is possible to experiment with different training parameters, such as optimization methods, variation of the learning rate, changing the number of iterations and epochs, among others.

Regarding the classification part, a possible development would be to omit the time restriction when classifying movements in real time. That is, to be able to collect data continuously without having to limit the collection to a specific period of time, and stop collecting them when the user indicates it.

Regarding the extension of ThingSpeak, a possible improvement would be the personalized sending of tweets for the user. That is, to be able to customize who the tweets are sent to so that they are directed to the desired user, and not simply to the application account.

Bibliografía

[1] Herrera PJ. Pajares G., B. and Besada E. *Aprendizaje Profundo*. RC-Libros, 2021.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[3] The Mathworks. Matlab drive, 2021. `https://es.mathworks.com/products/matlab-drive.html`.

[4] ThingSpeak. Thingspeak for iot projects, 2021. `https://thingspeak.com/`.

[5] Akram Bayat, Marc Pomplun, and Duc A Tran. A study on human activity recognition using accelerometer data from smartphones. *Procedia Computer Science*, 34:450–457, 2014.

[6] Ian H. Witten Eibe Frank, Mark A. Hall. *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.

[7] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Human activity recognition from accelerometer data using a wearable device. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 289–296. Springer, 2011.

[8] Jhon Ivan Pilataxi Piltaxi, María Fernanda Trujillo Guerrero, Vanessa Carolina Benavides Laguapillo, and Jorge Andrés Rosales Acosta. Human activity recognition using an accelerometer magnitude value. In *International Conference on Applied Technologies*, pages 462–472. Springer, 2019.

[9] Roberto Pavón Benítez. Técnicas de deep learning para el reconocimiento de movimientos corporales, 2020.

[10] The Mathworks. Matlab. deep learning toolbox, 2021. `https://es.mathworks.com/products/deep-learning.html`.

[11] Simon Kozina, Hristijan Gjoreski, Matjaž Gams, and Mitja Luštrek. Efficient activity recognition and fall detection using accelerometers. In *International competition on evaluating AAL systems through competitive benchmarking*, pages 13–23. Springer, 2013.

[12] Phataratah Sa-nguannarm, Ermal Elbasani, Bongjae Kim, Eung-Hee Kim, and Jeong-Dong Kim. Experimentation of human activity recognition by using accelerometer data based on lstm. In *Advanced Multimedia and Ubiquitous Engineering*, pages 83–89. Springer, 2021.

[13] Catherine Tong, Shyam A Tailor, and Nicholas D Lane. Are accelerometers for activity recognition a dead-end? In *Proceedings of the 21st International Workshop on Mobile Computing Systems and Applications*, pages 39–44, 2020.

[14] Daniel Ordóñez, Francisco Javier y Roggen. Redes neuronales convolucionales profundas y recurrentes lstm para el reconocimiento de actividad portátil multimodal. *Sensores*, (1), 2016.

[15] John McCarthy. What is artificial intelligence? 1998.

[16] PM GONZALO and PM Santos. Inteligencia artificial e ingeniería del conocimiento, ed. alfaomega. Technical report, 2006.

[17] Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014.

[18] Stuart J Russell and Peter Norvig. *Inteligencia Artificial: un enfoque moderno*. Number 04; Q335, R8y. 2004.

[19] Pedro Ponce Cruz. *Inteligencia artificial con aplicaciones a la ingeniería*. Alfaomega, 2011.

[20] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[21] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[22] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[23] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Google. Google colab., 2021. `https://colab.research.google.com`.

[26] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[27] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

ANEXO

En este anexo se proporcionan todos los detalles sobre cómo hacer uso de la aplicación desarrollada en Matlab a modo de guía de usuario.

Antes de ejecutar la aplicación es necesario disponer de unos requerimientos previos:

- Dispositivo móvil con la apliación Matlab Smartphone instalada.
- Conexión a Matlab Drive.
- Instalar el toolbox de *MATLAB Support Package for Android Sensors o MATLAB Support Package for Apple iOS Sensors*.

La aplicación ofrece tres pestañas principales:

- Inicio: En ella se da la bienvenida al usuario y se le informa del objetivo de la aplicación y de las diferentes funciones que podrá realizar.
- Entrenamiento de la red neuronal: En esta pestaña se le permite al usuario generar dos modelos de red distintos.
- Clasificación de acciones: Mediante el modelo de red que seleccione el usuario se mostrará la clasificación de los datos recogidos.

Todo el código fuente referente a este proyecto puede encontrarse en:

https://github.com/BNeku/TFG.git

Es necesario descargar el repositorio en la carpeta Matlab Drive.

A continuación se explica en profundidad cada apartado de la aplicación.

A.1 Entrenamiento de la red

Como se explica en el capítulo cuatro, se pueden entrenar dos tipos de redes, una red *default* o una red personalizada.

Red *default*

Para entrenar una red *default*, es necesario acceder a la pestaña **Entrenamiento red**, y pulsar el botón **Entrenar red** que se sitúa en la izquierda del panel de la aplicación (figura 4.2).
A continuación, tras esperar unos segundos, nos aparecerá el gráfico del proceso de entrenamiento.
Debido a que el archivo HumanActivityTrain contiene muchos datos, el entrenamiento puede tardar desde 2 horas hasta 4 horas.
Al terminar el proceso de entrenamiento, se guardará la con el nombre *redDefault* en la carpeta **redesEntrenadas** en el Matlab Drive.

Red personalizada

Esta red consiste en una red creada a partir de los movimientos elegidos por el usuario.
Para ello, previamente debemos capturar estos movimientos mediante los sensores de un dispositivo móvil utilizando la aplicación MATLAB para dispositivos móviles.

Recogida de movimientos propios

Para cada movimiento se realizará una recogida de datos. Para ello debemos activar todos los sensores de la aplicación Matlab Mobile. Para ello, accederemos desde el menú al apartado de sensores. Tras activar todos los sensores, también debemos cambiar la opción *Transmitir a* escogiendo la opción *Registro*. De esta forma, cuando captemos datos, estos se guardarán en un registro en una carpeta del Matlab Drive denominada *MobileSensorData*.

Para cada movimiento se debe crear un registro. Para ello se pulsa el botón Empezar de la aplicación MATLAB. Comenzando, acto seguido, a realizar el ejercicio que se desea captar. Al finalizar la acción, se pulsa el botón terminar. Al pulsar este botón, nos pedirá que introduzcamos el nombre del registro. Este nombre es el que se usará posteriormente para clasificar el movimiento, por lo que es conveniente nombrarlo como el propio movimiento que hemos realizado, como por ejemplo, sentado.

Según el tiempo de realización del ejercicio, más datos tendremos para su posterior clasificación, por lo que es aconsejable invertir tiempo en la recogida

de datos para que la red ajuste mejor sus pesos y alcance una mayor precisión con el menor error de pérdida posible.

Una vez que tenemos todos nuestros movimientos con datos en registros, se puede proceder con el entrenamiento.

En la pestaña **Entrenamiento red** se escribe, en el primer recuadro, el número de movimientos y por tanto acciones que identificará nuestra red y pulsamos el botón **Aceptar**. Si el número no es válido, no dejará pasar al siguiente paso.

Una vez introducido el número, deberemos escribir el nombre de los registros uno a uno, correspondientes a las acciones bajo entrenamiento. Tras escribirlo, primero se debe pulsar el botón **Buscar** para comprobar que este registro existe en la carpeta *MobileSensorData*, y si finalmente deseamos añadirlo para utilizarlo en el entrenamiento, deberemos pulsar el botón **Añadir**.

Una vez que hemos añadido tantos archivos como el número indicado en el primer paso, se habilitará el cuadro para elegir el nombre de la red que vamos a entrenar. Una vez introducido, pulsamos el botón **Aceptar**, y se habilitará el botón **Comenzar entrenamiento**.

Si se ha cometido algún error en alguno de los pasos, se habilita el botón **RESET** para iniciar el proceso desde el primer paso. Sin importar el momento en el que se pulse, siempre se volverá al primer paso para comenzar de nuevo el proceso.

Una vez que pulsamos el botón **Comenzar entrenamiento**, nos aparecerá una ventana en la cual se muestran los nombres de los registros introducidos. Si pulsamos **OK**, tras unos segundos se abrirá la ventana del proceso de entrenamiento, cuya duración dependerá de cuántos datos contienen los registros.

Si pulsamos **Cancel**, nos devuelve a la ventana anterior, de forma que podemos pulsar el botón **RESET** si queremos empezar de cero el proceso, o pulsar de nuevo el botón **Comenzar entrenamiento** si así lo deseamos.

Una vez finalizado el entrenamiento, se guardará una red con el nombre introducido anteriormente en la carpeta **redesEntrenadas** de Matlab Drive.

Este proceso también crea un archivo similar al de *HumanActivityTrain* utilizado en el entrenamiento de la *redDefault*. Este archivo se guarda con el nombre de la red, añadiendo *-Train* a este nombre, exactamente en la carpeta **archivosTrain** de Matlab Drive.

A.2 Clasificación de acciones

Como se ha explicado en el capítulo cuatro, es posible clasificar acciones mediante dos procedimientos distintos. A continuación, se explica cómo proceder en cada uno de ellos.

Captura y clasificación en tiempo real

Para poder clasificar acciones en tiempo real, la aplicación debe ejecutarse mediante Matlab Online con la conexión de Matlab Drive. De esta forma dispondremos del mismo espacio compartido con el dispositivo móvil. Es requisito fundamental activar todos los sensores del dispositivo accediendo desde la app móvil al **menú principal->sensores**, de no ser así, la captura de datos no estará disponible. Además, debemos configurar el sensor estableciendo la trasmisión a **MATLAB**.

Una vez establecida la configuración de los sensores, el usuario deberá elegir con qué modelo de red se van a clasificar los datos. Por defecto, la red utilizada será *redDefault*. Acto seguido, para comprobar que se puede establecer la conexión con el móvil se deberá pulsar el botón **Conectar**. Si existiera algún problema se notificaría al usuario mediante una ventana emergente. En cambio, si se ha podido cargar el modelo de red y la conexión con el dispositivo ha sido exitosa, la aplicación estará lista para iniciar la captura y clasificación de datos en tiempo real. Cuando el usuario esté listo para comenzar la acción se debe pulsar el botón **Comenzar captura de acciones**.

La captura comenzará automáticamente y tras unos segundos se detendrá, obteniendo en la interfaz los resultados de la clasificación.

Clasificación de acciones por archivos

En esta ocasión se ofrece al usuario importar su propio archivo de captura de datos. Al pulsar el botón **Importar** se abrirá un explorador de archivos donde el usuario podrá elegir el archivo que tenga guardado en el ordenador y desee clasificar. Si no existe ningún archivo y se desea capturar datos, el proceso se realiza mediante el dispositivo móvil, generando el resultado en la carpeta *MobileSensorData*, desde donde podrá importar dicho archivo.

Una vez seleccionado el archivo se iniciará el proceso automáticamente, obteniendo la moda de la clasificación por la interfaz.